UNIVERSITY OF CALIFORNIA
RIVERSIDE

Knowledge Discovery and Data Mining for Shared Mobility and Connected and Automated Vehicle Applications

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Chao Wang

March 2020

Dissertation Committee:
Dr. Matthew J. Barth, Chairperson
Dr. Guoyuan Wu
Dr. David Cocker

The Dissertation of Chao Wang is approved:

Committee Chairperson

University of California, Riverside

ACKNOWLEDGMENT

First and foremost, I want to sincerely thank my Ph.D. advisor, Dr. Matthew J. Barth. During the past five years I spent in the Transportation System Research (TSR) lab at UCR, he never hesitated to give me valuable advice not only for my research but also for my life-long career plan. He has been providing me a perfect environment to conduct research regarding my favorite topic and great opportunities to involve multiple well-funded projects. Besides his expertise in our research field, I also learned a lot from the way he communicates with people from different backgrounds, which I think will benefit me for my whole life. I would also like to thank Dr. David Cocker and Dr. Guoyuan Wu for their services in my dissertation committee. Specifically, I deeply appreciate Dr. Wu's patient and detailed guidance towards my dissertation, since many of my research topics originated from the brainstorms between us, and those topics cannot be transferred into so many solid publications without his help. He has also been willing to recommend me to the people he knows and help me expand my social network in our research field. My thanks also go to two other research faculty members in our lab, Dr. Peng Hao and Dr. Kanok Boriboonsomsin, for their management and guidance on the projects I'm involved in.

In addition to the aforementioned research advisors, my Ph.D. career cannot be successful without the help from so many people at UCR. I want to thank all my colleagues in our lovely TSR lab, including Mike Todd, Dr. George Scora, Alexander Vu, Dr. Ji Luo, Dr. Xuewei Qi, Dr. Danyang Tian, Dr. Fei Ye, Dr. Nigel Williams, Dr. Ziran Wang, Zhouqiao Zhao, Zhensong Wei, Xishun Liao, Yejia Liao, Yu Jiang and many more. I truly

appreciate my schoolmates and dear friends Yuanqi Gao for spending time with me working on different courses and projects.

Finally, I would like to acknowledge the Department of Electrical and Computer Engineering at UCR, as well as the College of Engineering-Center for Environmental Research and Technology (CE-CERT) at UCR, United States Department of Energy, United States Department of Transportation, National Center for Sustainable Transportation (NCST), National Renewable Energy Laboratory (NREL), for supporting my study and research in this dissertation.

ABSTRACT OF THE DISSERTATION

Knowledge Discovery and Data Mining
for Shared Mobility and Connected and Automated Vehicle Applications

by

Chao Wang

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, March 2020
Dr. Matthew J. Barth, Chairperson

The rapid development of shared mobility and connected and automated vehicles (CAVs) has not only brought new intelligent transportation system (ITS) challenges with the new types of mobility, but also brought a huge opportunity to accelerate the connectivity and informatization of transportation systems, particularly when we consider all the new forms of data that is becoming available. The primary challenge is how to take advantage of the enormous amount of data to discover knowledge, build effective models, and develop impactful applications. With the theoretical and experimental progress being made over the last two decades, data mining and machine learning technologies have become key approaches for parsing data, understanding information, and making informed decisions, especially as the rise of deep learning algorithms bringing new levels of performance to the analysis of large datasets. The combination of data mining and ITS can greatly benefit research and advances in shared mobility and CAVs.

This dissertation focuses on knowledge discovery and data mining for shared mobility and CAV applications. When considering big data associated with shared mobility

operations and CAV research, data mining techniques can be customized with transportation knowledge to initially parse the data. Then machine learning methods can be used to model the parsed data to elicit hidden knowledge. Finally, the discovered knowledge and extracted information can help in the development of effective shared mobility and CAV applications to achieve the goals of a safer, faster, and more eco-friendly transportation systems.

In this dissertation, there are four main sections that are addressed. First, new methodologies are introduced for extracting lane-level road features from rough crowdsourced GPS trajectories via data mining, which is subsequently used as the fundamental information for CAV applications. The proposed method results in decimeter level accuracy, which satisfies the positioning needs for many macroscopic and microscopic shared mobility and CAV applications. Second, macroscopic ride-hailing service big data has been analyzed for demand prediction, vehicle operation, and system efficiency monitoring. The proposed deep learning algorithms increase the ride-hailing demand prediction accuracy to 80% and can help the fleet dispatching system reduce 30% of vacant travel distance. Third, microscopic automated vehicle perception data has been analyzed for a real-time computer vision system that can be used for lane change behavior detection. The proposed deep learning design combines the residual neural network image input with time serious control data and reaches 95% of lane change behavior prediction accuracy. Last but not least, new ride sharing and CAV applications have been simulated in a behavior modeling framework to analyze the impact of mobility and energy consumption,

which addresses key barriers by quantifying the transportation system-wide mobility, energy and behavior impacts from new mobility technologies using real-world data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Shared mobility and connected automated vehicles (CAVs) are two key evolving directions of future transportation systems. For example, the ride sharing market has seen significant growth in recent years. A survey of almost 11,000 people in the U.S. indicated that 36% of people used ride sharing services in 2018, an increase from 15% in 2015. The largest two companies in the U.S. for ride hailing market are Uber and Lyft, which reported net revenue in 2018 of 11.3 billion U.S. dollars and 2.6 billion U.S. dollars respectively [1]. The worldwide revenue is expected to increase to 100.24 billion U.S. dollars by 2023, where ride hailing is one of the fastest-growing segments with an average annual growth rate of 14.8% [2].

With the emergence of self-driving cars and advanced driver assistance systems (ADAS), the self-driving car market has also become very active and extremely competitive. Although our roads are dominated by manually driving cars, that will soon change. The automated vehicle global market is expected to reach $36 billion by 2025, with North America owning 29% of all the self-driving vehicles in the world [3].

Further, with the rapid development of information and communication technologies, equipping automobiles with wireless communication capabilities is expected to be one of the biggest frontiers for automotive development. The market of connected vehicles is booming, and according to a recent business report, the global market is expected to reach 131.9 billion U.S. dollars by 2019 [4].

With the decrease of electronic devices size and cost, the number of sensors has greatly increased over the past two decades, which have been utilized in traffic monitoring infrastructure, on-board vehicle systems, and personal mobile devices. This trend has brought new data acquisition sources and deeply impacted the operation and evolution of Intelligent Transportation Systems (ITS). All participants of ITS act as data generators leading to a huge amount of available data with quick update rates. This growth in data production is being driven by: 1) individuals and their increased use of media (social networks); 2) novel types of sensors and communication capabilities in vehicle and the traffic infrastructure; and 3) application of modern information and communication technologies (cloud computing, Internet of Things, etc.). The proliferation of internet-connected devices and systems that generate ITS big data includes information-sensing mobile devices, aerial sensory technologies (remote sensing), software logs, cameras, microphones, radio-frequency identification readers, and wireless sensor networks, to name a few [5].

On the other hand, increasing traffic and frequent congestion in our modern transportation systems require innovative solutions for infrastructure and traffic management, as well as vehicle automation. Thus, there is an emerging need to utilize ITS

big data. In recent years, data-driven approaches for ITS applications have attracted significant research interest around the globe. However, since ITS big data is usually very large in size and contains a lot of noise, critical information is usually difficult to extract and summarize using typical data analysis techniques. Semi-automatic or automatic analysis of large quantities of data are required to extract previously unknown and interesting patterns, which is widely known as knowledge discovery and data mining.

By adopting and applying knowledge discovery and data mining techniques to ITS big data, particularly for shared mobility and CAVs, there will be large benefits to our transportation systems in some or all of the following aspects: 1) Travel time can be reduced through better operational strategies learned from historical ride sharing data; 2) The expense of road infrastructure mapping can be greatly reduced by exacting vehicle moving patterns from trajectory data; and 3) The safety and accuracy of automated driving and connected vehicles can be improved by learning driving behaviors from vehicle control and perception data.

1.2 Problem Statement and Contributions

The emerging challenges and opportunities brought by shared mobility and CAVs attracts significant research interests and has great potential to benefit the current transportation system in terms of safer, faster and more environment-friendly travel. An increasing amount of data has been generated for logging, monitoring, and perception purposes, and thereby contain valuable information and knowledge that can help accelerate this process. Therefore, it is essential to understand the ITS data associated with ride sharing and automated driving to help with the development of high-level applications.

Meanwhile, the development of data science and artificial intelligent has brought a wide variety of machine learning tools that could be widely used for offering insight and solutions in various engineering categories.

In this dissertation, a framework for knowledge discovery and data mining strategies has been developed and applied to facilitate shared mobility and CAV applications. The primary sources of data obtained from shared mobility and autonomous driving have been analyzed and mined for transportation knowledge for specific application development. The practicality of multiple data-driven methods for shared automated mobility applications are proven with several case studies outline in this dissertation. State-of-art machine learning methods have been customized and adopted to these case studies, providing good examples of the great potential of the crossover study of AI and ITS.

The main contributions of this dissertation can be summarized as:

- In order to offer road infrastructure information with timely and cost-efficient update for variety of shared mobility and CAV applications, a new data mining method has been developed for road infrastructure mapping that extracts intersection and stop bar locations purely from GPS position trajectories. The method includes a novel entropy-based analysis for intersection identification and a modified constrained Gaussian mixture model (CGMM) for stop bar position estimation. The proposed method results in decimeter level accuracy, which satisfies the positioning needs for many macroscopic and microscopic shared mobility and CAV applications.

- For macroscopic, a new data-driven framework has been developed for ride-hailing demand prediction and fleet operation optimization using data mining techniques applied to historical datasets. A Long Short-Term Memory (LSTM) network approach and a Convolutional Neural Network (CNN) model have been customized to predict short-term passenger demand, which helps solve the supply-demand optimal matching problem and improve the ride-hailing system efficiency.
- For microscopic, an end-to-end advanced driving assistant system was developed by learning driving behavior from video data. This helps with the decision making for autonomous driving vehicles so that they behave more like humans in terms of lane tracking. A case study of lane change behavior prediction and lane localization has been conducted, showing advances in practical accuracy and response time.
- To analyze the future impact of macroscopic and microscopic shared mobility and CAV applications, a comprehensive multi-agent transportation simulation was developed for shared mobility and CAV applications, utilizing resident's activity datasets. This simulation environment offers the insight of the future impacts that the shared and CAV applications would bring on the Southern California transportation system.

1.3 Dissertation Organization

The dissertation is organized as follows: Chapter 2 introduces the research background of this dissertation and conducts a literature review on several related topics, including different types of knowledge and data sources for modern ITS, the emerging

problems for shared automated mobility, and the state of art solutions with machine learning and data mining techniques. The following chapters covers three main aspects of the knowledge discovery and data mining for shared automated mobility: road infrastructure feature extraction, which is the key fundamental knowledge shared mobility and CAV applications; shared mobility demand estimation, representing macroscopic applications; and driver behavior prediction, representing microscopic applications. Specifically, Chapter 3 discusses knowledge discovery for road infrastructure by mining crowdsourced positioning data. Chapter 4 describes data driven demand prediction, dispatching operation, and system efficiency evaluation for ride hailing systems. Chapter 5 introduces end-to-end vision-based lane change behavior detection using deep learning. Finally, concluding remarks and future work are given in Chapter 6 which ties together the different components of the dissertation and provides insights into future the future directions for this work.

The flow of information from raw data to final applications for shared automated vehicles is shown in Figure 1. Ride sharing services and autonomous and semi-autonomous driving vehicles can generate many raw data sets, which are primarily used for sensing and monitoring purposes. In-depth information can be extracted from these raw data sets using data mining and machine learning methods for understanding the states of the vehicle, environment, and the overall transportation system. Applications based on the knowledge learned can then be developed with any necessary data fusion. Additional simulation and analytical research are conducted for the macroscopic and microscopic applications. As shown in the figure, the knowledge discovery from big data with be discussed in Chapter

3: road feature mapping by data mining crowdsourced trajectories, Chapter 4: Data-Driven Ride-Hailing Demand Prediction, and Chapter 5: Deep Learning based Realtime Computer Vision System for Lane Change Behavior Detection. Chapter 6 conducted a comprehensive multi-agent transportation simulation with BEAM for shared mobility and CAV applications, utilizing resident's activity datasets to analyze the future impact of macroscopic and microscopic shared mobility and CAV applications.

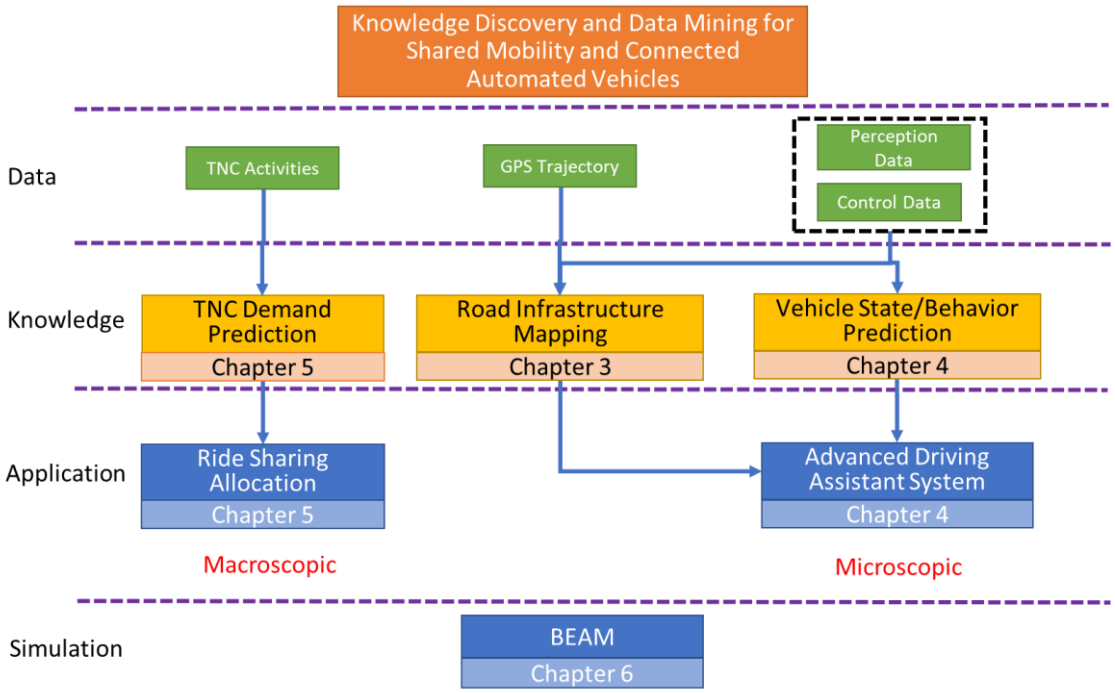


Figure 1. Architecture for knowledge discovery and data mining for shared mobility and vehicle automation applications.

Chapter 2

Research Background and Literature Review

2.1 Data Mining for Applications in ITS

Data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in dataset. This process helps in extracting and refining useful knowledge from large datasets. The extracted information can be used to form a prediction or classification model, identify trends and associations, refine an existing model, or provide a summary of the datasets being mined. Numerous data mining techniques of various types such as rule induction, neural networks, and conceptual clustering have been developed and used individually in domains ranging from space data analysis to financial analysis [6]. A recent review by Kohavi [7] states that data mining serves two goals namely Insight and Prediction. Insight leads to identifying patterns and trends that are useful. Prediction leads to identifying a model that gives reliable prediction based on input data.

In the field of transportation engineering, large amounts of data are generated during studies on traffic management, accidents analysis, pavement conditions, roadway feature inventory, traffic signals and signal inventory, bridge maintenance, road

characteristics inventory, among other things. Based on these data, decision-makers arrive at decisions to solve a respective problem. Decision-makers are always on lookout for ways to ease the pain in obtaining access to and applying disparate datasets. The basic requirements include the ability to identify what data is available, determine the characteristics of the data, extract the data of interest, and transform the data into formats necessary for the application. In a real-life transportation domain situation, diverse fields of data need to be collected to integrate and to arrive at the solutions. Recent research study in the field of data mining approach has opened a new horizon for decision-makers of transportation engineers.

There is a broad spectrum of engineering problems where computational intelligence is becoming an essential part in many advanced systems. Such problems arise in data processing, which is faced with huge data explosion, due to automatic data collection systems and the possibility for combining data from many sources over data networks. Hence new techniques for extracting important knowledge from the raw data are required to efficiently handle such data. In the areas of intelligent transportation research, data mining is broadly used for road infrastructure information extraction, macroscopic shared mobility data analysis, and microscopic automated and connected vehicle input processing. This thesis will illustrate how these three categories of applications are developed by knowledge discovery and data mining with study instances: lane-level road feature mapping with crowdsourced GPS trajectories, ride-haling demand prediction with fleet activities, and lane change behavior prediction with image data mining.

2.2 Data Mining for Road Feature Mapping with Lane-Level Accuracy

The recent development of shared mobility and connected and CAVs is producing increasing demand on digital maps with detailed road features which are required for many applications, such as lane-level navigation and intersection movement assistance. In addition, this information usually needs continuous updates to ensure its accuracy and reliability for safety purposes. The state of art technique to survey detailed road features is to use the Light Detection and Ranging (LiDAR) sensors, which can conduct 3D point-cloud registration of the target by illuminating the target with pulsed laser light to correct the global localization error and achieve centimeter-level accuracy [8], [9], [10]. Although this approach generates a digital map with lane-level details, the cost is still too high [11] for mass adoption and the data processing is computationally expensive in terms of both space and time complexity[12].

To achieve higher map updating efficiency with low cost, mining road features from probe vehicle data attracts more and more research attention recently. The increasing number of vehicles equipped with sensors provides large-scale data and enables information extraction techniques to map road features. It has been proven that the Global Positioning System (GPS) data can help to build road-level maps [13]. Early works like [27], [28], [29] introduced methods to generate road level map information with GPS trajectories. For the last two decades, generating more detailed lane level road features with vehicle trajectories raised extensive research interests. However, it is still challenging to extract more detailed features such as stop bars and lane markers, because both positioning errors and driver behaviors introduce noises and uncertainties to the data sources. Thus,

more advanced data filtering and mining methods, along with statistical analysis and modeling, are required in feature extraction to improve the estimation accuracy. Wilson et al. introduced the idea that uncoordinated probe vehicles with positioning capabilities and communications could be used to map the lane network with potential decimeter accuracy [30]. In the following work [63], a data-mining approach was developed to build a lane model that gave predictions with high accuracy from a small number of trips passing over a certain road segment. The main idea is to cluster the traces for different lanes and use the mean of each cluster as the lane centerline (represented by the offset to road centerline). Similarly, the Kernel Density Estimation method [72] was designed to estimate the probability density function (PDF) of the vehicles' positions on a cross-section. With the PDF, the lane centerlines are well represented by the peaks of the function. This method is independent of the lane width and lane parallelism and can handle lane splits and merges. In [73][30], Gaussian Mixture Model (GMM) was adopted to cluster the GPS trajectories in the lateral direction and identify the positions of centerlines based on the means of clusters.

2.3 Data Mining with Ride-hailing Service Activities

For macroscopic, in recent years, ride-hailing services provided by transportation networking companies (TNCs) such as Uber [31], Lyft [32], DiDi Chuxing [33], and RideAustin [34] are emerging as a new and disruptive on-demand mobility services. Accurate prediction of TNC trip demand not only enable city to better understand TNC trip patterns and proactively control traffic, but also help TNCs and drivers make informed decisions to minimize deadhead vehicle miles traveled (VMT) and reduce impact of TNC

on congestion and energy consumptions. In the next wave of transportation innovations, such as mobility-as-a-service (MaaS) where on-demand shared autonomous vehicles (SAVs) transport people and goods in cities, accurate trip demand prediction will provide decision making tools for automated vehicle fleet operators to optimize SAV routes for the whole city.

Considerable interests in predicting the trip demand for taxi or TNC trips have grown in the research community for the last a few years. Chang et al. [35] mined historical data to predict taxi demand distributions using clustering algorithms. Moreira-Matias et al. [36][37] applied time series techniques to forecast taxi passenger demand. Gong et al. [38] proposed a machine learning model, XGBoost, to predict New York City Taxi demand. While the majority of the work was conducted for taxi trip demand, the studies on TNC trip demand prediction are relatively limited. Recently, Ke et al. [39] introduced the fusion convolutional long short-term memory network (FCL-Net) to forecast passenger demand for on-demand ride services in Hangzhou, China, using real-world data provided by DiDi Chuxing. Wang et al. [40] developed a Long Short-Term Memory (LSTM) to predict the number of Uber pickups in the City of New York. Xu et al. [41] also developed a LSTM to predict taxi passenger demand for each small area in New York City. Liao et al. [42] conducted a thorough comparison between two deep neural network structure, ST-ResNet and FLC-Net, for taxi demand prediction using New York City taxi data. They found out that deep neural networks outperform most traditional machine learning models when predicting taxi trip demand.

Most of the previous studies adopted time-series model or recurrent neural network architecture, which are good at capturing time dependencies from historical data. One drawback of these models is that spatial information is usually lost in the modeling process. This paper proposes a convolutional neural network (CNN) based deep learning model to predict the TNC trip demand considering both temporal and spatial features. CNN is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery and proven to be a very efficient and well-performed image recognition algorithm. CNN was inspired by biological processes [43]. The connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use a variation of multilayer perceptron designed to require minimal preprocessing [44]. Similarly, the convolutional layers of a CNN not only enable CNN to capture the local feature of the image data, but also reduce model parameters that need to be estimated. A time-space trip demand matrix can be constructed by spatial-temporal trip demand data. Then, CNN is able to learn trip demand as images and make predictions. In Chapter 4, different algorithms a tested for ride-hailing demand predictions and novel deep learning methods based on LSTM and CNN are proposed and evaluated.

2.4 Data Mining for Vision Based Lane Change Detection

For microscopic, data mining and machine learning technologies are more and more used in automated vehicles perception tasks, both at an academic and industrial level.

The goal in each case is to arrive at a full understanding of the environment around the car using various sensors and control modules. Camera-based perception is an important step towards such environmental understanding as it allows the car to properly position itself within the road lanes. It is also crucial for any subsequent lane departure or trajectory planning decision. As such, performing accurate camera-based lane detection in real-time is a key enabler of fully autonomous driving. Traditional lane detection methods [45][46] rely on a combination of highly specialized, handcrafted features and heuristics to identify lane segments. Popular choices of such hand-crafted cues include color based features [47], the structure tensor [48], the bar filter [49], ridge features [50], etc., which are possibly combined with a Hough transform [51][52] and particle or Kalman filters [53][54][55]. After identifying the lane segments post-processing techniques are employed to filter out misdetections and group segments together to form the final lanes. For a detailed overview of lane detection systems, we refer the reader to [56]. In general, these traditional approaches are prone to robustness issues due to road scene variations that cannot be easily modeled by such model-based systems. More recent methods have replaced the hand-crafted feature detectors with deep networks to learn dense predictions, i.e. pixel-wise lane segmentations. Gopalan et al. [57] use a pixel-hierarchy feature descriptor to model contextual information and a boosting algorithm to select relevant contextual features for detecting lane markings. In a similar vein, Kim and Lee [58] combine a Convolutional Neural Network (CNN) with the RANSAC algorithm to detect lanes starting from edge images. Note that in their method the CNN is mainly used for image enhancement and only if the road scene is complex, e.g. it includes roadside trees, fences, or intersections. Huval

et al. [59] show how existing CNN models can be used for highway driving applications, among which an end-to-end CNN that performs lane detection and classification. He et al. [60] introduce the Dual-View CNN (DVCNN) that uses a front-view and a top-view image simultaneously to exclude false detections and remove nonclub shaped structures respectively. Li et al. [61] propose the use of a multi-task deep convolutional network that focuses on finding geometric lane attributes, such as location and orientation, together with a Recurrent Neural Network (RNN) that detects the lanes. Most recently, Lee et al. [62] show how a multi-task network can jointly handle lane and road marking detection and recognition under adverse weather and low illumination conditions. Apart from the ability of the networks to segment out lane markings better, their big receptive field allows them to also estimate lanes even in cases when no markings are present in the image. At a final stage, however, the generated binary lane segmentations still need to be disentangled into the different lane instances. In Chapter 5, a novel end-to-end deep learning strategy are studied for lane change detection with image data mining.

Chapter 3

Road Feature Mapping by Data Mining Crowdsourced Trajectories

3.1 Introduction

Digital road feature mapping is one of the most important and challenging tasks for intelligent transportation systems. One of the traditional methods is to use satellite images or aerial images [15], [16], [17][17]. However, up-to-date satellite images are not always available to the general public, and aerial images are expensive to update. Even though the camera resolution is rapidly increasing, it is still too limited to facilitate the generation of a lane-level map from satellite/aerial images [18].

Over the past several years, fully autonomous, self-driving cars have grown into a reality with progress in the simultaneous localization and mapping research community. The navigation system and other applications for autonomous cars require precise localization within an a priori known map of high accuracy. State-of-the-art methods use reflectivity measurements from LiDAR scanners to create an orthographic map of ground-plane reflectivity (3D point cloud data). Accurate road information can be extracted from point cloud data captured by airborne **[**19] and mobile LiDAR system (MLS) [20][20]. For instance, Yadav et al. proposed a method to extract rural road surface using LiDAR point

cloud data [21]; Lehtomäki et al. used vehicle-based LiDAR scanning data to detect vertical pole-like objects in road environments [22]; Liu et al. developed a new curb detection method for autonomous vehicles LiDAR data [23].

With the development of data science, position data has been widely used for transportation research. P. Hao et al. proposed a framework to evaluate environmental impact by traffic congestions using mobile positioning data [24]. C. Wang et al. introduced data-driven methods for ride-hailing system demand prediction using pickup and drop-off positioning data. Positioning information can also be used for road feature extraction [25], [24]. Prob vehicles equipped with GPS sensors can help infer the road network when the trajectory data is analyzed.

While many research efforts have been conducted on mapping lateral road features, i.e. lane separators or lane centerlines of the roadways, the longitudinal features such as stop bars at intersections received insufficient attention and investigation. To our best knowledge, there have been very few studies aiming at estimating stop bar positions. In addition, most previous studies were based on prior knowledge of existing road-level maps, which limits their applicability. To achieve higher map updating efficiency with low cost, mining road features from probe vehicle data attracts more and more research attention recently. The increasing number of vehicles equipped with sensors provides large-scale data and enables information extraction techniques to map road features. It has been proven that the Global Positioning System (GPS) data can help to build road-level maps [13]. However, it is still challenging to extract more detailed features such as stop bars and lane markers, because both positioning errors and driver behaviors introduce noises and

uncertainties to the data sources. Thus, more advanced data filtering and mining methods, along with statistical analysis and modeling, are required in feature extraction to improve the estimation accuracy. In this chapter, we proposed a framework that can identify intersections and extracts stop bar positions by only using vehicle positioning trajectories collected from connected vehicles. The main contributions of this work are:

- Designed a framework to extract detailed road features from vehicle positioning trajectories without prior geographic knowledge.
- Proposed a novel method to identify intersections by analyzing the vehicle moving direction entropy.
- Defined the constrained Gaussian mixture model (CGMM) and derived the formulas for the customized expectation-maximization (EM) algorithm to estimate the stop bar position.
- Achieved meter-level accuracy for stop bar position estimation which can be used for most of the mobility and environmental oriented connected and automated vehicle applications such as eco-approach and departure at signalized intersections [14].

3.2 Intersection Region Identification by GPS Heading Entropy Analysis and Stop Bar Position Estimation using Gaussian Mixture Model

3.2.1 System Architecture

The overall process of intersections identification and stop bar localization are provided in Figure 2.

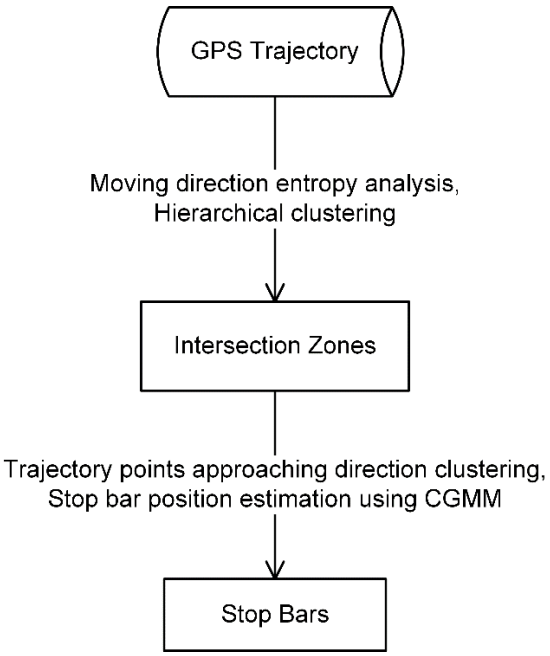


Figure 2. Intersection and Stop Bar Position Estimation System architecture.

The method of estimating stop bar positions is composed of two main steps.

1. Intersection identification.

- A study region is first selected to determine the range of the urban area of interests. The GPS trajectories inside the study region are selected as the observation data.
- Then the study region is discretized into square cells and an entropy analysis method is used to determine the cells located at intersections. The main idea of the method is: vehicle trajectories inside intersections would have a higher diversity of moving directions than those inside other regions.

- Last, a hierarchical clustering method is used to group the nearby intersection cells together into different intersection zones.

2. Stop bar position estimation.

- In each intersection zone, hierarchical clustering is used to cluster GPS trajectories into different approaching directions.
- For each approaching direction, the stop bar positions are then estimated by modeling the distribution of the data points using the constrained Gaussian mixture model (CGMM).
- Please note that the input of this method is only the vehicle GPS trajectories and it does not require any prior geographic knowledge.

3.2.2 Intersection Identification

The intersection zones can be identified with four steps: 1) discretize the study region into cells; 2) analyze the GPS heading entropy of the trajectory in each cell; 3) select cells with entropy high entropy by threshold; 4) group the selected cells that are closed to each other to form intersection zones. The methodology applied for these steps as well as the considerations behind the ideas is illustrated as follows.

Compared to the driving on non-intersection road segments, vehicles at intersection areas have more diverse moving directions. Therefore, intersections can be identified by selecting areas with a high diversity of moving directions. A good index to measure the moving direction diversity is the Shannon entropy (or Shannon index) [75].

In the GPS logged data, the moving direction of a vehicle is recorded as GPS heading. For a specific region on the road segment, the GPS heading entropy for all the trajectory points inside the region can be calculated by the following equation:

$$H = -\sum_{i=0}^{n} p_i \log P_i \qquad (1)$$

where n is the number of angular intervals and p_i is the proportion of heading angle belonging to the *i*-th angular interval. The radar chart in Figure 3 shows an example of the GPS heading distribution, the total number of data points is 1189 and the number of heading angle between 300 and 330 is 203, therefore the proportion of vehicle driving within that direction range is 0.17.

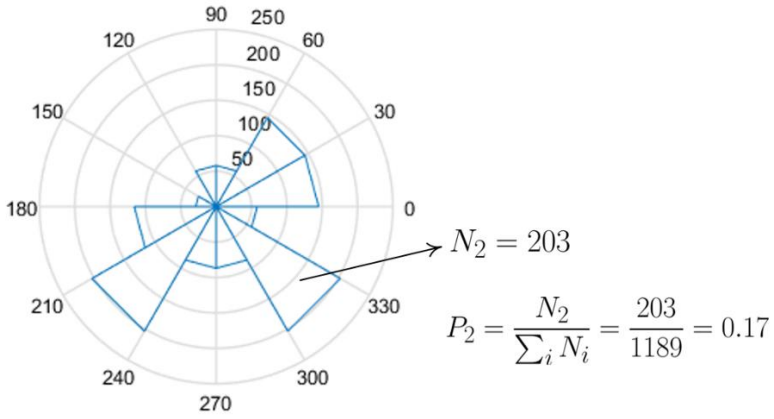


Figure 3. Example of GPS heading distribution in one cell and the calculation of the entropy.

To study the GPS heading entropy in an urban area, the 2-dimensional space can be discretized into square cells. Within each cell, the entropy of GPS heading from the trajectory data can be calculated. The cells within an intersection area would have higher entropy values compared to the cells on other parts or off the road segments. Figure 4 visualizes an example of the GPS heading entropy for the regular road segment, intersection area, and non-road area. As shown in the figure, for the areas with no trajectory

covered, the GPS heading entropy would be zero; for a regular road segment (Cell B), the GPS heading diversity is low compared with intersection area thus it has relatively low entropy; and the intersection area (Cell A) has higher entropy because of the high moving direction diversity of the trajectories.

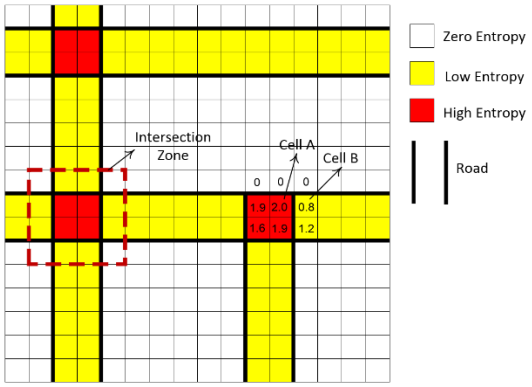


Figure 4. Example of the GPS heading entropy for regular road segment (yellow cells), intersection area (red cells) and non-road area (white cells). The numbers inside cells indicate the value of GPS heading entropy.

Cells with GPS heading entropy higher than threshold h_e are selected as intersection cells, which will be further clustered so that nearby cells that belong to the same intersection are grouped together to form intersection zones. The method used for high entropy cell clustering is the hierarchical clustering method [76]. It is commonly used when the number of clusters is unknown but the dissimilarities limit between clusters are more certain. Strategies for hierarchical clustering generally fall into two types:

- Agglomerative: 1) Assign each observation to its own cluster; 2) Compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters; 3) Repeat steps 2) until there is only a single cluster left.

- Divisive: 1) Assign all the observations to a single cluster and then partition the cluster to two least similar clusters; 2) Recursively conduct the same procedure on each cluster until each cluster has only one observation.

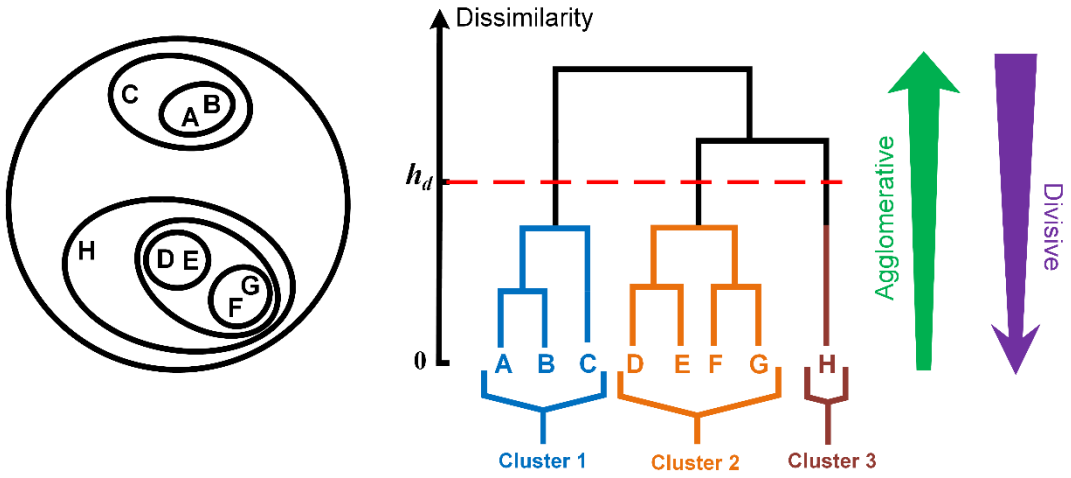


Figure 5. Hierarchical clustering illustration. The left side is the elements clustering hierarchy based on the distance (similarities). The right side is the dendrogram, which can be generated by either argumentative or division manner of the hierarchical clustering algorithm. h_d is the dissimilarity threshold to determine clusters.

The output of hierarchical clustering (by either agglomerative or division) is a dendrogram, which shows the hierarchical relationship between the clusters (Figure 5). Every leaf in the dendrogram carries one observation. As we move up, the leaf observations begin to merge into nodes (carrying observations that are similar to each other). Lower the merging happens (towards the bottom of the tree), more similar the observations will be. Higher the merging happens (toward the top of the tree), less similar the observations will be. To determine clusters, a dissimilarity threshold h_d makes horizontal cuts across the branches of the dendrogram. The clusters are then generated by the separated branches.

Algorithm 1 Hierarchical Clustering

Given:
  Set X of observations $\{x_1, \dots, x_n\}$
  Distance function $\mathbf{dist}(c_1, c_2)$
for $i = 1$ *to* n
  $c_i = x_i$
end for
$C = \{c_1, \dots c_n\}$
while $C.size > 1$ do
  $(c_{min1}, c_{min2}) = \mathbf{argmin}\left(\mathbf{dist}(c_i, c_j)\right) \; \forall \, c_i, c_j \in C$
  if $dist(c_{min1}, c_{min2}) < h_d$ // h_d **is the dissimilarity threshold**
    Break
  end if
  $C = C - c_{min1} - c_{min2}$
  $C = C \cup \{c_{min1}, c_{min2}\}$
end while

Algorithm 1 describes the hierarchical clustering in the agglomerative manner. A distance function is required to determine the distance between each cluster, which is defined by the linkage criteria and the measure of similarity (distance). The single-linkage is used in this study: the distance between two clusters is defined as the shortest distance between two points in each cluster, which can be expressed by:

$$\text{dist}(c_1, c_2) = \min\left(\text{D}_\text{e}\left(x_{c_1 i}, x_{c_2 j}\right)\right), \forall x_{c_1 i} \in c_1, x_{c_2 j} \in c_2 \quad (2)$$

where D_e is the Euclidean distance and works as the measure of the distance of two observations. The Euclidean distance in two dimensions can be expressed as:

$$\text{D}_\text{e}(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \quad (3)$$

where $\boldsymbol{p} = (p_1, p_2), \boldsymbol{q} = (q_1, q_2)$ are two points in the Euclidean plane.

3.2.3 Stop Bar Position Estimation

Within each intersection zone, the stop bar position can be estimated by three steps: 1) extract the stationary points from the trajectories in the intersection zone; 2) cluster the stationary points for each approaching direction; 3) estimate the stop bar position using CGMM for each approaching direction. The methods details are further introduced by the following explanation and mathematical derivations.

Stationary points are defined as the GPS data point collected when the vehicle is stopped (i.e. GPS speed is zero). The GPS heading information for these points are extracted for driving direction detection. To avoid extra noise caused by vehicles that make turns at intersections, only stationary points of straight trajectories within the determination area are used for further clustering analysis.

Within the marked intersection zone, the selected stationary points are grouped into different driving directions by hierarchical clustering since the number of stop bars (4 for a 4-way intersection and 3 for a 3-way intersection) is unknown. We use the hierarchical clustering method mentioned above with the same type of metric and linkage criteria. The grouped stationary points are then applied to further estimate the actual position of the stop bar for each direction.

The stop position of a vehicle at an intersection is highly dependent on the vehicle's position in the queue. A vehicle can either stop at the stop bar or stop behind preceding vehicles in the queue. Therefore, we build a stochastic model to estimate vehicle stop positions with stationary points as input observations. Since the stop bar is a longitudinal traffic control marking, its longitudinal position (position along the road) is the most

concern. Correspondingly, the stop position is defined as the vehicle's longitudinal position along the road.

Let P_1 be the measured stop position of the first vehicle in a queue which stops at the stop bar. It is a random variable which is a combination of two factors: 1) the distance between the GPS sensor and the stop bar; and 2) the GPS error of the equipped sensor. A proper assumption is that P_1 is Gaussian distributed: $P_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)\mathrm{P_1 \sim N(\mu_1, \Sigma_1)}$. Also assume the space headway (distance between the heads of two consecutive vehicles in a queue) is a Gaussian random variable:$S \sim \mathcal{N}(\mu_s, \sigma_s^2)\mathrm{S \sim N(\mu_s, \Sigma_s)}$, which is the sum of the preceding vehicle's length and the gap in between. Thus, the *k*-th stop position from the stop bar is

$$P_k = \begin{cases} P_1 \quad , & k = 1 \\ P_1 + \sum_{j=1}^{k-1} S_j \quad , & k \geq 2 \end{cases} \qquad (4)$$

where $S_1, S_2, \mathrm{L}\ , S_{k-1}\ \mathrm{S_1, S_2, S_{k\text{-}1}}$ are independent and identically distributed (IID) random variables with the distribution of $\mathcal{N}(\mu_s, \sigma_s^2)\mathrm{N(\mu_s, \Sigma_s)}$.

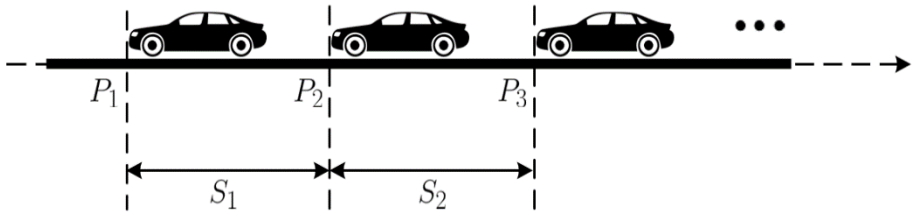


Figure 6. Vehicle stop position modeling.

The measured position is the GPS error $\mathrm{W_{gps}}$ W_{gps} added up to the actual position

$$P_k{}' = P_k + W_{gps} = \begin{cases} P_1 + W_{gps} \quad , & k = 1 \\ P_1 + W_{gps} + \sum_{j=1}^{k-1} S_j, & k \geq 2 \end{cases} \qquad (5)$$

For modeling convenience, we assume the GPS noise to be white Gaussian $W_{gps} \sim \mathcal{N}(0, \sigma_{gps}^2)$ [77]. Since P_1, S_j, W_{gps} $\mathrm{S_1, S_2, S_{k\text{-}1}}$ are independent with each other, according to Gaussian distribution's property [78], [79], the sum $P_k \mathrm{X_k}$ is also in Gaussian distribution$\mathrm{P_k} \sim \mathrm{N}(\mu_\mathrm{k}, \Sigma_\mathrm{k})$ $P_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$, with its mean being the sum of the means of P_1, S_j, W_{gps}, and the square of its standard deviation is the sum of the squares of the standard deviations of P_1, S_j, W_{gps}:

$$\mu_k = \mu_1 + (k-1)\mu_s \qquad (6)$$

$$\sigma_k^2 = \sigma_1^2 + \sigma_{gps}^2 + (k-1)\sigma_s^2 \qquad (7)$$

Therefore, along the road, the actual position of a stationary point is a mixture of all *m* components with mixture rate π_k $\mathrm{\pi_k}$:

$$X = \sum_{k=1}^{m} \pi_k P_k \ , \quad \sum_{k=1}^{m} \pi_k = 1 \qquad (8)$$

The actual stop bar position can be acquired by estimating the parameters of this constrained GMM.

EM algorithm for parameter estimation

Consider an estimation problem with $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)\mathrm{x} = (\mathrm{x_1, x_2, \cdots, x_n})$ being the *n* observations, and let $\boldsymbol{z} = (z_1, z_2, \cdots, z_n)\mathrm{z} = (\mathrm{z_1, z_2, \cdots, z_n})$ be the unobserved latent variable. The goal is to fit the parameters $\boldsymbol{\theta}$ of a statistic model $\mathrm{p(x,z)}p(\boldsymbol{x}, \boldsymbol{z})$ to the data, where the likelihood function is given by

$$L(\boldsymbol{\theta}; \mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}) = \sum_z p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) \qquad (9)$$

It can be done with maximum likelihood estimation (MLE), which is formulated by the following equation:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{argmax}\{log(L(\boldsymbol{\theta}; \boldsymbol{x}))\} \qquad (10)$$

However, it is hard to find the MLE by taking gradient because of the latent variable. In this case, the EM algorithm gives an efficient method to solve MLE, which iteratively constructs a lower-bound on the likelihood function (E-step) and optimizes that lower-bound (M-step) until convergence. The EM algorithm consists of two steps:

- ***Expectation step (E-step)***: Calculate the expected value of the likelihood function, with respect to the conditional distribution of $\mathbf{z}$ given $\mathbf{x}$ under the current estimated $\boldsymbol{\theta}^{(t)}\theta^{(t)}$:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = E_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\theta}^{(t)}}[log\, L\,(\boldsymbol{\theta}^{(t)}; \boldsymbol{x}, \boldsymbol{z})]$$

$$= E_{\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\theta}^{(t)}}[log\, p\,(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}^{(t)})] \qquad (11)$$

- ***Maximization step (M-step)***: Find the parameter that maximizes this quantity:

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{argmax}\{Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})\} \qquad (12)$$

Reference [80] gives the proof that the EM algorithm forces the likelihood to converge monotonically. Therefore, the parameters of statistic models with unobserved latent variables can be estimated in an iterative manner.

However, for the CGMM we defined, the parameters we aim to estimate are not independent for each Gaussian component, meaning conventional EM algorithm needs to be customized for the constraints indicates by equation (6), (7). For the customized EM algorithm, the parameters being estimated are $\boldsymbol{\pi} = [\pi_1, \pi_2, \cdots, \pi_m]\pi = [\pi_1, \pi_2, \cdots, \pi_m]$,

$\mu = [\mu_1, \mu_2, \cdots, \mu_m]$ $\boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_m]$ and $\boldsymbol{\Sigma} = [\sigma_1^2, \sigma_2^2, \cdots, \sigma_m^2]\Sigma = [\Sigma_1, \Sigma_2, \cdots, \Sigma_m]$, expressed as $\boldsymbol{\theta} = [\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}]\theta = [\pi, \mu, \Sigma]$. The algorithm is derived as follows.

- **Customized EM Algorithm**
  - ***Modified E-step***: Given our current estimate of the parameters $\boldsymbol{\theta}^{(t)}\theta^{(t)}$, the conditional distribution of $\boldsymbol{z_i}$ (membership probabilities) for the *k*-th Gaussian component is

$$W_{k,i}^{(t)} = p\left(z_i = k|x_i; \boldsymbol{\theta}^{(t)}\right) = \frac{\pi_k^{(t)} p\left(x_i; \mu_k^{(t)}, \left(\sigma_k^{(t)}\right)^2\right)}{\sum_{j=1}^{m} \pi_j^{(t)} p\left(x_i; \mu_j^{(t)}, \left(\sigma_j^{(t)}\right)^2\right)} \quad (13)$$

Fixing $W_{k,i}^{(t)}$, the Q Q function is constructed as below:

$$Q\left(\boldsymbol{\theta}\middle|\boldsymbol{\theta}^{(t)}\right) = E_{\left(\boldsymbol{z}\middle|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}\right)}\left[log\, p\left(\boldsymbol{x}, \boldsymbol{z}\middle|\boldsymbol{\theta}^{(t)}\right)\right]$$

$$= E_{\left(\boldsymbol{z}\middle|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}\right)}\left[\sum_{i=1}^{n} log\, p\left(x_i, z_i\middle|\boldsymbol{\theta}^{(t)}\right)\right]$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{m} w_{k,i}^{(t)}\, log\, \pi_k^{(t)} \; +$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} w_{k,i}^{(t)}\, log\, p\left(x_i; \mu_k^{(t)}, \left(\sigma_k^{(t)}\right)^2\right) \quad (14)$$

As defined above, the density of *k*-th Gaussian component is

$$p\left(x_i; \mu_k^{(t)}, \left(\sigma_k^{(t)}\right)^2\right)$$

$$= \frac{1}{\sqrt{2\pi\left(\sigma_k^{(t)}\right)^2}} exp\left(-\frac{\left(x_i - \mu_k^{(t)}\right)^2}{2\left(\sigma_k^{(t)}\right)^2}\right)$$

$$= \frac{1}{\sqrt{2\pi\left(\sigma_k^{(t)}\right)^2}} exp\left(-\frac{\left(x_i - \mu_1^{(t)} - (k-1)\mu_s^{(t)}\right)^2}{2\left(\left(\sigma'^{(t)}_1\right)^2 + (k-1)\left(\sigma_s^{(t)}\right)^2\right)}\right)$$

$$= f_k\left(x_i, \mu_1^{(t)}, \mu_s^{(t)}, \left(\sigma'^{(t)}_1\right)^2, \left(\sigma_s^{(t)}\right)^2\right) \quad (15)$$

where $\left(\sigma'^{(t)}_1\right)^2 = \left(\sigma_1^{(t)}\right)^2 + \left(\sigma_{gps}^{(t)}\right)^2 \Sigma_1^{'(t)} = \Sigma_1^{(t)} + \Sigma_{gps}^{(t)}$. Substitute to the Q function, we have

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{n}\sum_{k=1}^{m} w_{k,i}^{(t)} \, log \, \pi_k^{(t)} +$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} w_{k,i}^{(t)} \, log \, f_k\left(x_i, \mu_1^{(t)}, \mu_s^{(t)}, \left(\sigma'^{(t)}_1\right)^2, \left(\sigma_s^{(t)}\right)^2\right) \quad (16)$$

- ***Modified M-step***: To maximize $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ $Q\left(\theta|\theta^{(t)}\right)$, we can maximize the term containing $\pi_k^{(t)}$ and the term containing f_k since they are not related. Thus, the closed form of the maximum of π_k is

$$\pi_k^{(t+1)} = \underset{\substack{\pi_k \\ subject\ to: \sum_{k=1}^{m}\pi_k = 1}}{argmax} \left\{Q\left(\boldsymbol{\theta}\middle|\boldsymbol{\theta}^{(t)}\right)\right\} = \frac{1}{n}\sum_{i=1}^{n} w_{k,i}^{(t)} \quad (17)$$

To find the argument maxima of the rest parameters is a 5-dimensional optimization problem and hard to derive. So we introduce the expectation conditional maximization (ECM) algorithm [81][81]. ECM is a variant of EM algorithm, which replaces each M-step with a sequence of conditional maximization (CM) steps in which each parameter is maximized individually, conditionally on the other parameters remaining fixed. Therefore, we have:

$$\frac{\partial}{\partial \mu_1} Q\left(\boldsymbol{\theta}\middle|\boldsymbol{\theta}^{(t)}\right) \equiv 0 \quad (18)$$

$$\frac{\partial}{\partial \mu_s} Q\left(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}\right) \equiv 0 \qquad (19)$$

$$\frac{\partial}{\partial \Sigma_1'} Q\left(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}\right) \equiv 0 \qquad (20)$$

$$\frac{\partial}{\partial \Sigma_s} Q\left(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}\right) \equiv 0 \qquad (21)$$

Equation (18) and (19) can be solved analytically for the maximized parameters:

$$\mu_1^{(t+1)} = \frac{\sum_{i=1}^{n}\sum_{k=1}^{m}\left(x_i-(k-1)\mu_s^{(t)}\right)\left(\left(\sigma'^{(t)}_1\right)^2+(k-1)\left(\sigma_s^{(t)}\right)^2\right)^{-1} w_{k,i}^{(t)}}{\sum_{i=1}^{n}\sum_{k=1}^{m}\left(\left(\sigma'^{(t)}_1\right)^2+(k-1)\left(\sigma_s^{(t)}\right)^2\right)^{-1} w_{k,i}^{(t)}} \qquad (22)$$

$$\mu_s^{(t+1)} = \frac{\sum_{i=1}^{n}\sum_{k=1}^{m}\left(x_i-\mu_1^{(t)}\right)(k-1)\left(\left(\sigma'^{(t)}_1\right)^2+(k-1)\left(\sigma_s^{(t)}\right)^2\right)^{-1} w_{k,i}^{(t)}}{\sum_{i=1}^{n}\sum_{k=1}^{m}(k-1)^2\left(\left(\sigma'^{(t)}_1\right)^2+(k-1)\left(\sigma_s^{(t)}\right)^2\right)^{-1} w_{k,i}^{(t)}} \qquad (23)$$

It is hard to find analytical solutions for equation (20). However, further extending the ECM algorithm, the iteration method can still get a converged solution by only seeking an increase in the objective function in the M-step instead of a strict optimum [82][82].

Based on the above customized EM algorithm, we will be able to acquire the estimation of the first stop position in the queue right at the stop bar:

$$E[P_1] = \mu_1 \qquad (24)$$

We define b as the bias that the first stop position shifted from the stop bar position. It would be influenced by the driving behavior and the GPS antenna equipped position. The stop bar longitudinal position can be expressed as:

$$Y = \mu_1 + b \qquad (25)$$

We can calibrate b using part of the labeled data, i.e., manually calculating the average bias for some randomly sampled stop bars.

3.3 Experiment Setup, Result and Analysis

In this section, the proposed methods are validated by applying to the data from the Safety Pilot Model Deployment (SPMD) program [83] in Ann Arbor, Michigan. The results are analyzed and presented in this section.

3.3.1 Dataset

The SPMD program is a part of the Connected Vehicle Safety Pilot Program funded by the U.S. Department of Transportation (USDOT). The main objectives of the SPMD include the exploration of the real-world effectiveness of connected vehicle safety applications in multi-modal driving conditions, the evaluation of how drivers adapted to the use of this connected vehicle (CV) technology, and the identification of potential safety benefits as a result of CV technology. The SPMD data environment contains sanitized mobility data elements that were collected from over 2700 vehicles, equipped with connected vehicle technologies, traversing the Ann Arbor, MI transportation network. The vehicle data were also collected from roadside equipment installed along with the transportation network via vehicle-to-infrastructure (V2I) technology. Both datasets were collected in two separate months, October 2012 and April 2013.

The data in this research were derived from the onboard wireless safety unit (WSU), consisting of GPS-based data elements and those that were obtained from the vehicle's Controller Area Network (CAN) Bus. In addition to GPS-based data, there are a series of data elements that indicate vehicle performance information and the state of a few of its

components. In this study, only the GPS data was used for analysis and it is worth mentioning that the GPS trajectory was contributed by the Ann Arbor citizens in a crowdsourced manner. The University of Michigan invited citizens to attend the research and contribute data from their day-to-day travel activities. In order to protect the privacy of SPMD participants, the published document along with the data are void of data elements that either contain (Sensitive) Personally Identifiable Information ([S]PII) or that may lead to the discovery of PII. The data used for this study has been pre-processed for privacy issues by the publisher. There are 27 fields in each data file. The parameters used in this study listed in Table 1.

Table 1. Parameters used in the Safety Pilot Dataset.

Field Name	***Units***	***Description***
Device	N/A	Unique numeric ID
Trip	N/A	Count of ignition cycles.
Frequency	0.1s	Time in centiseconds since DAS started.
GpsValidWsu	N/A	Communicates GPS validity.
GpsTimeWsu	0.01s	Epoch GPS time received from WSU.
LatitudeWsu	degree	Latitude from WSU receiver.
LongitudeWsu	degree	Longitude from WSU receiver.
AltitudeWsu	m	Altitude from WSU receiver.
GpsHeadingWsu	degree	Heading from WSU GPS receiver.
GpsSpeedWsu	m/sec	Speed from WSU GPS receiver.

3.3.2 Identify the Intersection zones

The study region is in the urban area of Ann Arbor near the campus of the University of Michigan. The boundary longitudes and latitudes are west: -83.7532°; east: -83.7281°; north: 42.2845°; south: 42.2739°, respectively. Figure 8 shows the study region, a rectangular area that is1.2 km from north to south and 2 km from west to east. The study region contains 94 intersections (including 3-way and 4-way intersections) and 332 stop bars.

Following the methods proposed in section III, the study region is discretized into cells and the GPS heading entropy in each cell is calculated. Figure 8 shows part of the area and the entropy values are marked by color. Figure 9 shows the heat map of the whole study region. As expected, the cells at intersections have higher entropy than the ones on regular road segments. Too large or too small of the cell size would not work well. Because too large of the cell size would result in low spatial resolution and gives inaccurate intersection center. Too small of the cell size would result in higher computation cost and if the size goes to extremely small, there would not be enough trajectories in each cell for analysis. We select the cell size based on the entropy contradiction defined as:

$$\Delta H = \min H_{zone} - \max H_{nonzone} \qquad (27)$$

Where H_{zone} represents the entropy for the cells within intersection zones, the $H_{nonzone}$ is the entropy for cells outside of intersection zones. The entropy contrast ΔH reflects how well the intersection cells are highlighted among the surrounding cells. Figure 7 shows the sensitivity analysis for the cell size and the entropy contrast, which indicates that the intersection is the most distinguishable when the cell size is 8 m. Correspondingly,

the entropy contrast is 1.8 at 8 m of cell size, which can be used as the threshold h_e mentioned in Section II.B to filter intersection cells.

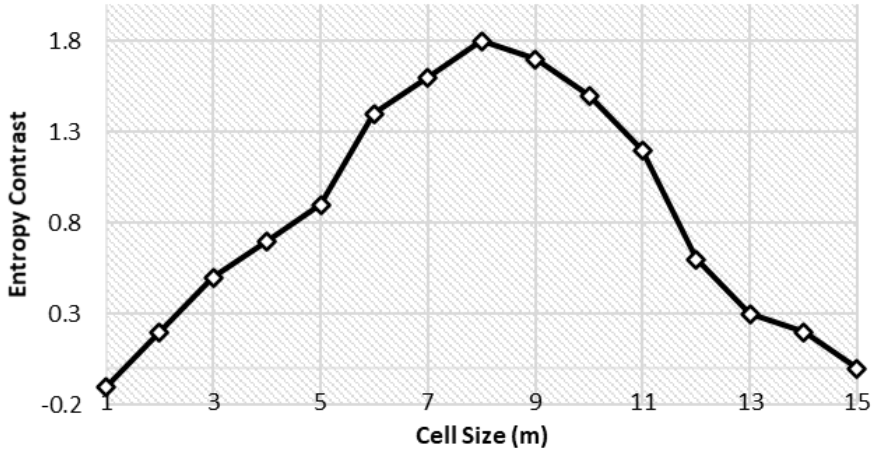


Figure 7. Cell size vs. GPS heading entropy contrast.

Algorithm 1 is then used to group the selected cells to form intersection zones. The linkage threshold h_d is swept from 1 m to 10 m to find the optimized value for the best specification of each intersection. Table 2 concludes the parameters used for intersection identification.

Table 2. Parameter Specified for Intersection Identification.

Notation	*Description*	*Value*
a	Cell side length for study region discretization.	8 m
h_e	The threshold to filter cells with high GPS heading entropy.	1.8
h_d	Linkage threshold for cell clustering.	5 m

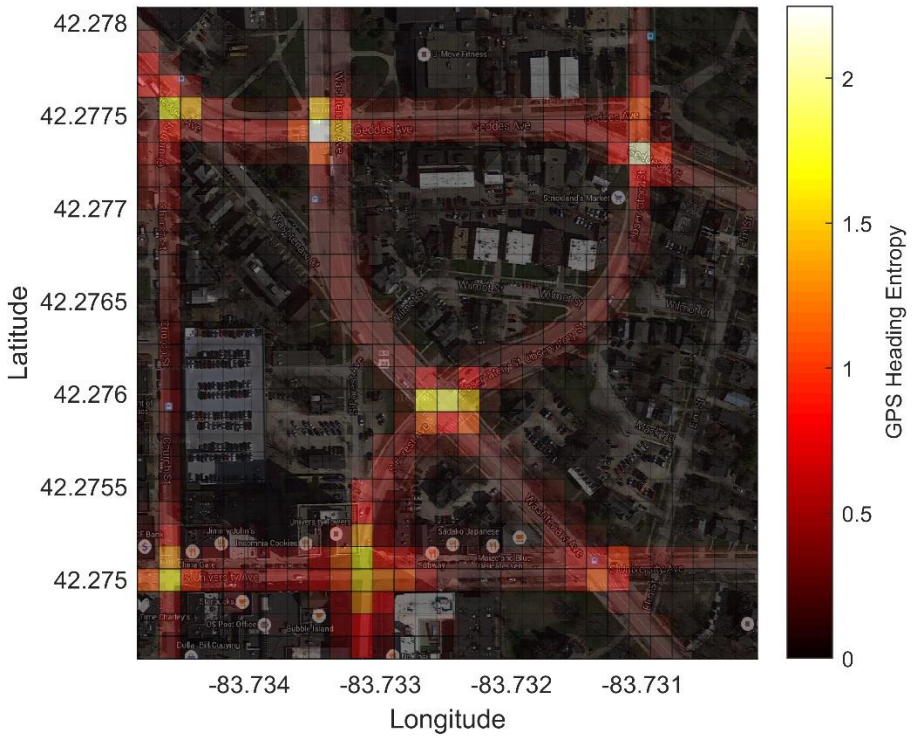


Figure 8. GPS heading entropy for cells in different regions. Cells within intersections have higher entropy than others.

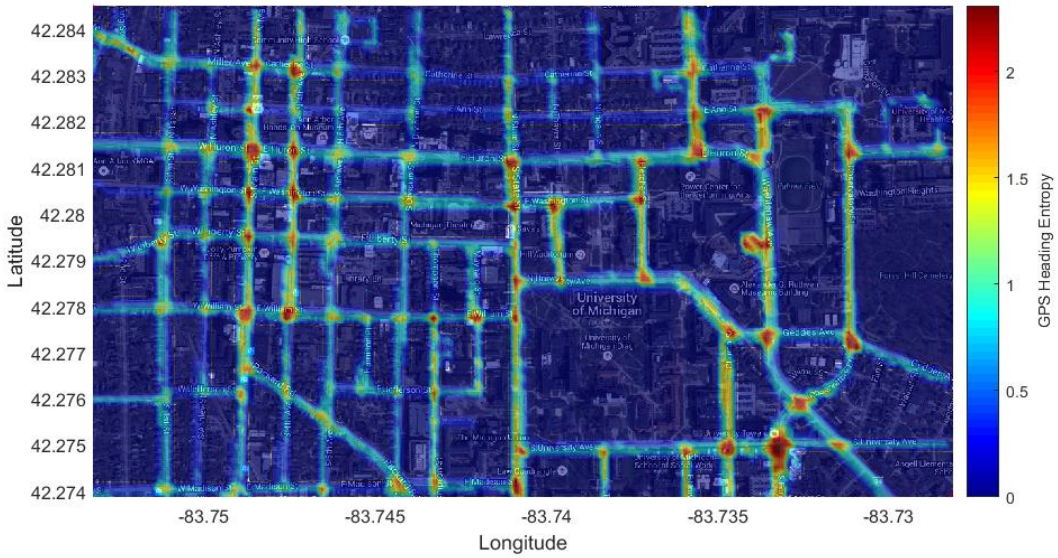


Figure 9. GPS heading entropy heat map.

The GPS heading entropy is calculated according to equation (1). In the figure, red indicates high entropy and blue indicates low entropy. As expected, the intersection areas with a higher diversity of vehicle moving directions higher entropy.

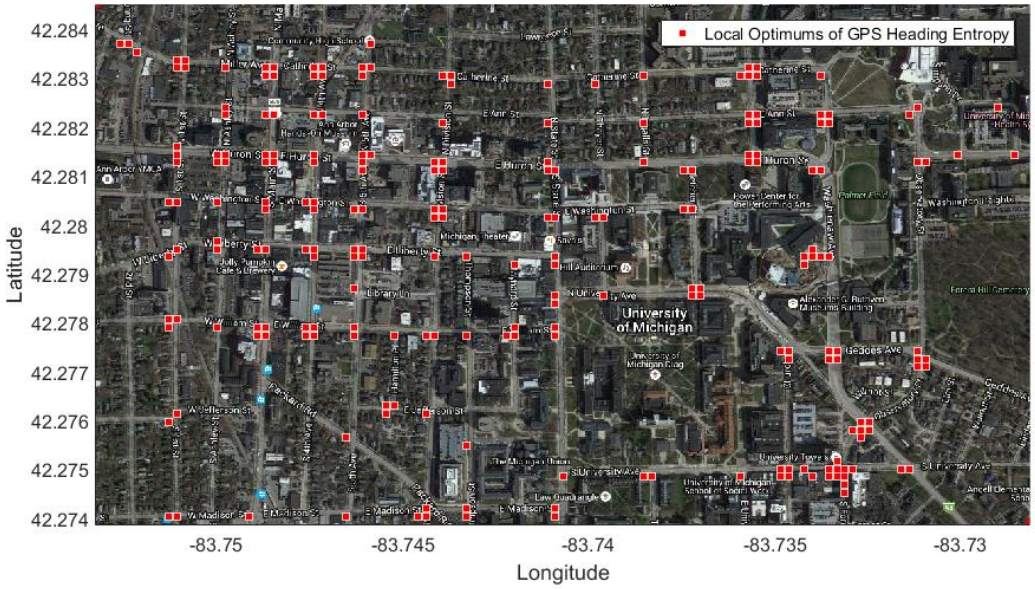


Figure 10. Filtered cells with high GPS heading entropy.

After clustering, we mark each intersection using the weighted cluster central, which is calculated as the average position of the inside cells weighted by the entropy value:

$$x_{ci} = \sum_{j=1}^{N_i} H_j x_j \qquad (24)$$

$$y_{ci} = \sum_{j=1}^{N_i} H_j y_j \qquad (25)$$

In order to include more data samples for stop bar detection and to handle the intersection positioning error, extended square areas are used to represent the intersection zones. As shown in Figure 10, most intersections are accurately identified, while some other intersections are not detected as there is too little traffic passing through them (e.g.,

less than 2 trajectories). For the 94 intersections covered by trajectories of GPS-equipped vehicles within the study region, the accuracy of intersections marking is 95.7%. The average error on intersection center estimation is 12 meters, which is accurate enough for the succeeding analysis. Some other regions with high driving direction entropy such as parking lots are mistakenly marked. But they can be easily removed by analyzing the speed profile from the GPS trajectories.

Figure 11. Identified Intersection zones. The regions framed by white squares are the identified intersections. The center of each intersection zone is marked with a green cross.

3.3.3 Stop Bar Number and Locations Determination

The intersection zones are framed by 70 m × 70 m squares. The side length of the square is selected to include enough upstream trajectories. In this study case, a side length from 50 m to 80 m performs well in the stop bar position extraction step. To identify the

number and location of stop bars, we choose the intersection of East Huron Street and Division Street as an example. Before we applied the approach direction and stop bar position estimation algorithm to the intersection, we filtered and process the raw data from the intersection zone as follows:

(1) Speed filtering. The stationary points in the trajectories are selected. Considering the error and lag effect of GPS sensor, we not only select data points with strictly zero speed, but also data points with very low speed. The threshold is set to be 0.1 m/s.

(2) Approach clustering. We utilize hierarchical clustering to group these stationary points into each approaching direction based on GPS headings. The clustering process starts with each point as a cluster, and then we keep merging clusters until the minimum linkage of any two clusters is smaller than the linkage threshold, which we set as 20° in our case study.

A boundary problem will arise if we directly apply a clustering algorithm to the GPS headings. For example, GPS headings of 355° and 5° are both near northward with 10° as the angular distance, but they will not be grouped together as the clustering algorithm would simply consider the angles as scalars and the difference$|355° - 5°| > 20°|355°\text{-}5°| = 350° > 20°$. This problem can be solved by using vectors to express GPS headings:

$$\vec{v} = (cos\, h\, , sin\, h) \tag{26}$$

where h is the GPS heading. Then the 2-D vector $\vec{v}\overrightarrow{v}$ is used as the feature for clustering. As shown in Figure 11, three approaches are clearly classified based on the

heading vector, which indicates that the proposed method works perfectly without any miss-grouping.

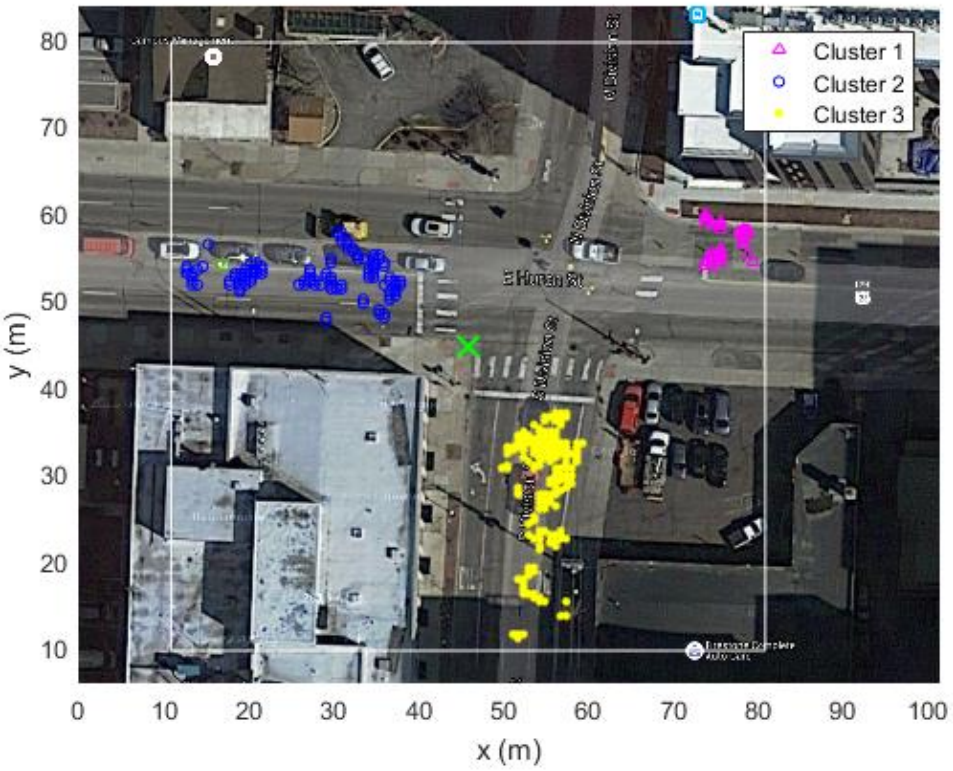


Figure 12. Identified approaches directions at the intersection.

For the filtered data of each approach, we apply the proposed method in section III to estimating the location of the corresponding stop bar. Here, we carry out the proposed method with approach 3 (cluster 3 in Figure 11) in the last step as an instance. All the stationary points' coordinates are projected to the approaching direction. The histogram of the distance to an origin point is shown in Figure 12.

Before searching the optimal solution for the proposed Gaussian mixture model, we set constraints for the parameters to increase the computation efficiently and conceptual

validity. These constraints are set by taking the boundary values of the prior knowledge. For example, the space headway is always greater than the minimum possible car length and smaller than the maximum possible car length plus a large clearance. The number of Gaussian components is limited to 3, as the GPS position of the 4th vehicle (or vehicles behind) in a queue has little correlation with the stop bar position. The constraints are summarized in Table 3. The estimated distribution is shown in Figure 12.

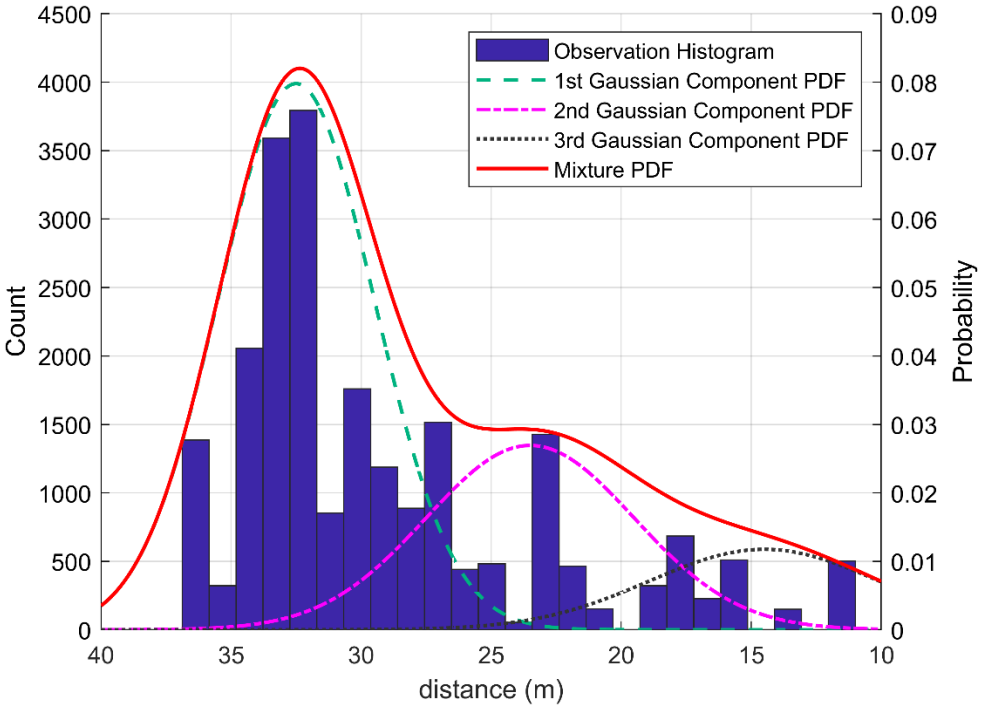


Figure 13. Estimated PDF for the data point.

Table 3. Parameter Constraints of the CGMM.

Parameters	***Description***	***Constraints***
μ_s	Mean of the space headway	$5 \leq \mu_s \leq 10$
σ_s	The standard deviation of the space headway	$0 \leq \sigma_s \leq 1.5$
σ_{gps}	The standard deviation of the 1-dimensional GPS error	$0 \leq \sigma_{gps} \leq 2$
π_1	Mixture rate of first Gaussian	$0.3 \leq \pi_1 \leq 1$
π_2	Mixture rate of second Gaussian	$0 \leq \pi_2 \leq \pi_1$
π_3	Mixture rate of third Gaussian	$0 \leq \pi_3 \leq \pi_2$

Applying the method mentioned in III, we calibrate the bias from the first stop position to the stop bar position using part of the dataset as a training set. We have

$$b = 0.46\ m \tag{28}$$

Figure 13 visualizes the results for estimated stop bars. The stop bar midpoint coordinates along the road are calculated based on the CGMM parameter estimation above. The stop bar directions are orthogonal to the average moving direction of each approach. And the length of the stop bar is determined by the limits of lateral positions of the trajectory at the stop bar. The same method is applied to the other 93 intersections in the study region. Including the intersection studied above, we generate the position of 332 stop

bars. To evaluate the accuracy of the results, we sued the stop bar positions on Google Earth as ground truth.

Figure 14 shows how we compare our results with the ground truth. The red line represents the ground-truth stop bar and the green one represents the estimated stop bar. We define the error metrics as follows:

1) Angular error $\Delta\theta$, which is the acute angle between the actual stop bar and the estimated one.

2) The longitudinal projection error, which is the projection distance between the actual and estimated stop bars' midpoints along the road direction. We project the midpoints of the two bars to the road direction. y is the coordinate of the actual stop bar projection and $\hat{y}$is the coordinate of the estimated stop bar projection. We define $\Delta y = |y - \hat{y}|$is the longitudinal projection error.

Figure 14. Estimated stop bar positions.

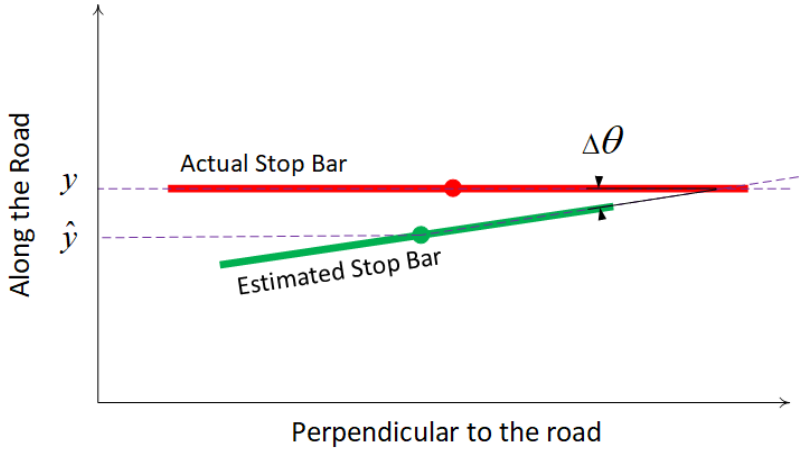


Figure 15. Stop bar error measurements.

Table 4. Parameter Constraints of the GMM.

Error	*Mean*	*Standard Deviation*
Angular error $\Delta\theta$	3.7°	1.9°
Longitudinal projection error Δy	0.27 m	0.32 m

Table 4 shows the mean and the statistics results of the two kinds of errors above. We find that the mean and standard deviation of longitudinal position errors are 0.27m and 0.32m respectively. That means the accuracy of the stop bar detection is in meter-level, which will satisfy most of the mobility and environmental oriented connected and automated vehicle applications such as eco-approach and departure at signalized intersections [84]. By the time of this dissertation being written, there are very few researches on the stop bar position estimation using GPS data. Therefore, it is hard to find a benchmark for result comparison. We thus developed a trivial baseline for evaluation of the proposed method. For the trajectories in each approach of a intersection, the average position with the minimum moving speed are calculated, which can be formulated as:

$$\hat{y} = \frac{1}{n}\sum_{i \in n} \underset{y \in T_i}{\operatorname{argmin}}\, v_y \qquad (29)$$

Table 5 shows the accuracy comparison of the baseline and the proposed method. Since many factors are not considered in the baseline model, like the upstream stop and go caused by congestion, it performs significantly worse. Mean absolute error (MAE) is used for the comparison.

Table 5. Model Comparison.

Model	*Longitudinal projection error (MAE)*
Baseline	4.8 m
Proposed model	0.3 m

3.4 Conclusions and Discussion

This study proposes a data-driven approach to detect stop bars at intersections. This method is capable of efficiently extracting stop bar positions of all the intersections in a given area by mining the data of enough vehicle GPS trajectories without any prior map information. The dataset from the Safety Pilot Model Deployment Program is applied for testing. Numerical analysis shows that 95.7% of the intersections covered by trajectories within the test region is correctly marked with an average shift of 12 m comparing to the ground truth. For stop bar positioning, the mean and standard deviation of the error are 0.27 m and 0.32 m, respectively. The accuracy can satisfy most of the mobility and environmental oriented connected and automated vehicle applications such as eco-approach and departure at signalized intersections. The future work will involve an investigation of the impact of different road geometry designs on the stop bar positioning. And these differences can be potentially learned from data. Considering the additional information (space headway) learned by the CGMM estimation process, we may take advantage of this information in the future study to better understand the traffic conditions or driving behaviors at intersections in a specific urban area. Furthermore, the entropy analysis method can be used for freeway traffic information mining to find disordered and chaotic spots caused by accidents or heavy traffic.

Chapter 4

Data-Driven Ride-Hailing Demand Prediction

4.1 Introduction

Over the last two decades, the development of sensing and communication technology has brought the vehicles to a more connected and intelligent level. In particular, widely equipped positioning systems (GPS, GSM, etc.) offer us a new source of spatial-temporal data of running vehicles. Using data mining and machine learning techniques and discovering knowledge from this kind of data is becoming a very popular research topic. Taxicabs are one of the earliest types of vehicles equipped with positioning systems and collect trip data in a large scale. There are many available taxi data and many interesting studies. The first type of applications utilizing the taxi data is for traffic monitoring and prediction. Min and Wynter [85] developed a method that provides predictions of speed and volume over 5-min intervals for up to 1 h in advance of the traffic. This method considers the spatial characteristics of a road network in a way that reflects not only the distance but also the average speed on the links. Lv et al. [86] presented a deep learning method with stacked auto-encoders model to predict the traffic. This is the first work apply deep learning for traffic prediction and the authors showed the proposed method

outperforms many traditional machine learning methods. Another application of taxi data is planning. Chen et al. [87] proposed a method to plan the night bus route by looking at the taxi pickups at night. Spatial clustering and optimized routing are used in their method. Zhan et al. [88] researched the taxi data in New York City and tried to obtain the theoretical optimum of the taxi allocation by using optimal matching of the taxicabs and passengers. Their results showed a large gap between the current taxi service system and the theoretical maximum, indicating great room for improvement.

The number of taxi pickups itself reflects the demand of the ride service, which is also valuable to model. Moreira-Matias et al. [91] proposed a hybrid method to predict the number of taxi pickups at a certain taxi stop. Three models are utilized and combined to get the best guess. Besides traditional taxicabs, on-demand, app-based ride services like Uber and Lyft has become an important part of today's transportation system with its flexibility and quick responsiveness. While some data analysis and visualization work have been illustrated [92], [93], few studies are on-going to utilize the Uber-like taxi data. Uber-like taxis all have loggers to monitor and record trip information such as pickup location and trip distance, which can be a valuable data source for knowledge discovering. Compared with traditional taxis, Uber-like taxis are directed to the passengers and one request of ride service always correspond to a pickup. Therefore, Uber-like taxi pickups better reflect the true ride service demand. So, it is more meaningful to predict the number of pickups of Uber like taxis since a better understanding and forecasting of the travel demand helps to improve the efficiency and sustainability of the urban transportation system. And newly aroused applications like ride sharing and autonomous mobility

dispatching are based on solid demand predictions. In this chapter, we proposed a deep learning method to predict the number of Uber pickups with long short-term memory (LSTM) model. The LSTM model is compared with a statistic-based baseline: time-varying Poisson model, and a basic machine learning model: regression tree model. The results show the proposed method outperforms the baselines and the improvement is significant.

4.2 Methodology

Consider the number of Uber pickups overtime in a certain region of interest is a discrete time series $\{X_0, X_1, \dots, X_t\}$. The problem is formulated as: given $\{X_0, X_1, \dots, X_{t-1}\}$, build a model to determine X_t. For this study, the time step of the series is one hour. Figure 1 shows a sample sequence of the number of Uber pickups over time at Midtown Center, Manhattan. We can clearly observe the periodic pattern of the pickup sequence. To solve the proposed prediction problem, three models are introduced in this section: time-varying Poisson model, regression tree model, and the long short-term memory (LSTM) model. The time-varying Poisson model and the regression tree model are introduced as baselines for comparison purpose. Our deep learning approach is based on the LSTM model.

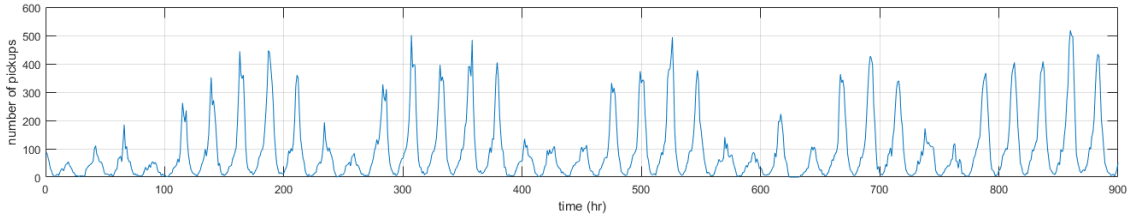


Figure 16. Example sequence of Uber pickups.

4.2.1 Time-Varying Poisson Model

This baseline model was proposed in [91]. The pickup demand pattern reflects human activity and appear to be non-homogeneous. Assume the probability for n Uber pickups to emerge in a certain time slot follows a Poisson distribution:

$$y(n;\lambda) = \frac{e^{-\lambda}\lambda^{n}}{n!}$$

where λ is the expectation of the number of pickups in this time slot. λ is a time varying parameter, i.e., $\lambda(t)$, since the average pickups varies with time. If we consider the pickup sequence is periodic and the period is one week, $\lambda(t)$ can be defined as the following equation, where $\lambda(0)$ is the average pickups over a full week, $\delta_{d(t)}$ and η $d(t), h(t)$ are the correction factors for with respect to day of week and hour of day, respectively. Consider$\lambda(t)$ as a discrete function (hourly aggregated).

$$\lambda(t) = \lambda(0)\delta_{d(t)}\eta_{d(t),h(t)}$$

$$\sum_{d=1}^{7} \delta(d) = 7$$

$$\sum_{h=1}^{24} \eta(d,h) = 24 \ \forall\, d$$

4.2.2 Regression Tree Model

Decision tree learning is a widely used predictive modeling approach in statistics, data mining and machine learning. Regression trees are the kind of decision trees whose target variables can take continuous values. Regression tree-based learning model builds a relationship between input variables and the targets by categorizing targets with boundary conditions of the inputs. A tree can be "learned" by splitting the source set into subsets based on an attribute value test [94]. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. See the examples illustrated in the figure for spaces that have and have not been partitioned using recursive partitioning, or recursive binary splitting. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees (TDIDT) is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data. Figure 2 shows an example of regression tree construction [95] of X_1, X_2,, are the two dimensions of the input feature. The output $Y \in \{R_1, R_2, \dots, R_5\}$ is function of X_1, X_2, which can be express by the tree on the left. Every node of the tree corresponds to a split decision, and each leaf contains a subset of the data that satisfies the conditions. Therefore, to predict a new data point with input feature x_1, x_2, we can use the following equation, where $f(x_1, x_2)$ is the regression tree model and hm is the indicator of regions (For example, region R_1: $h_1 = I(x_1 \leq t_1)I(x_2 \leq t_2)$).

$$\hat{y}_t = f(x_1, x_2) = \sum_{m=1}^{M} \beta_m h_m(x_1, x_2)$$

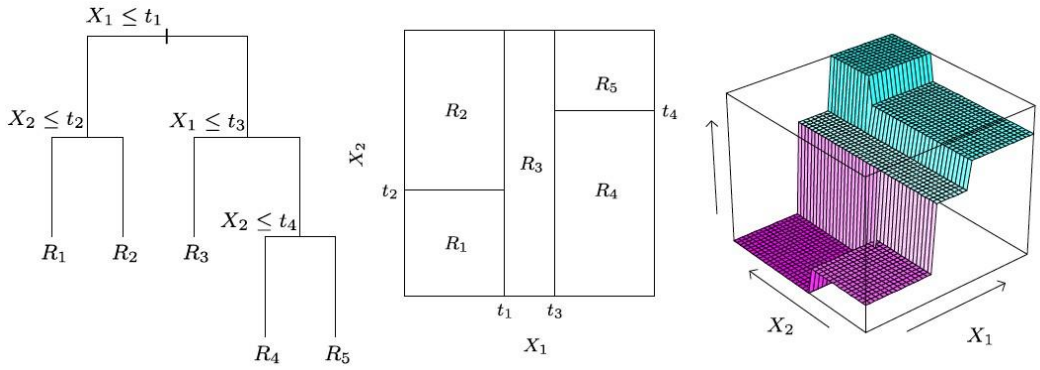


Figure 17. Regression tree example [11].

For the number of demand prediction problem. We can select proper features as the input train a regression tree to predict the future number of pickups. From Figure 1, the pickups are showing weekly and daily periodic patterns. And the current hour pickups should be related to the previous hour's, considering the temporal correlation. Also, the pickups may be affected by the weather condition. Table 6 are the selected feature for the regression tree model.

Table 6. Features used for regression tree model.

attribute type	Symbol	data
Features	x_1	Day of week
	x_2	Hour of day
	x_2	number of pickups of current hour
	x_2	precipitation of the day
	x_2	temperature of the day
output	y	number of pickups of next hour

4.2.3 LSTM Model

For our pickup demand prediction problem, we saw that the number of pickups is not only dependent on short-term data of near past but also present periodic patterns and tight dependencies on long-term histories. In other words, the correlation between two data point in the sequence is not necessarily monumentally decreasing as time difference increases. To better present the long-term dependencies (repeating patterns), we calculated the autocorrelation of a sample pickup sequence. Autocorrelation is the correlation of the sequence with a delayed version of itself. Let X be the pickup sequence, X_s be the subsequence start at time s, and X_t be the subsequence start at t. Suppose the mean and

variance of the two sequences are μ_s, μ_t, σ_s, σ_t respectively, then the autocorrelation between time s and t is:

$$R(s,t) = \frac{E[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s}$$

where E is the expected value operator. We took $s = 0$ and calculated the autocorrelation $A(t) = R(0,t)$. Figure 3 shows the result. We find that the number of pickups at a certain time of one day is highly correlated with the number of pickups of the same time on other days especially on the same day of other weeks.

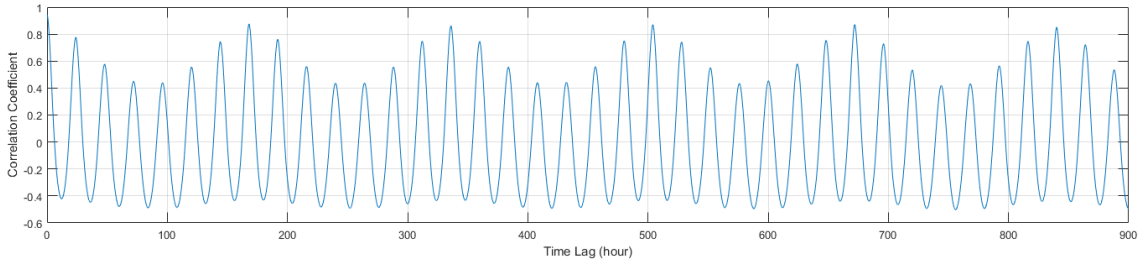


Figure 18. Autocorrelation of the pickups sequence.

Recurrent neural networks (RNN) can capture the long-term dependencies reflected by Figure 3. They are artificial neural networks with a loop in them, allowing information to pass on to succeeding steps. Hence, to let the networks to "memorize" former inputs. Figure 4 presents an example RNN and its unfolding in time of the computation involved in its forward computation. x_t is the input at time step t, and s_t is the hidden state at time step t. s_t is the function of s_{t-1} and x_t: $s_t = f(W_{s_{t-1}} + U_{x_t})$. O_t is the output at time step t. Despite the theoretical capability of memorizing previous information in the network, RNN still suffers to learn the connections between inputs as the time gap in between grows in practice. This shortcoming of RNN is further explained by [96].

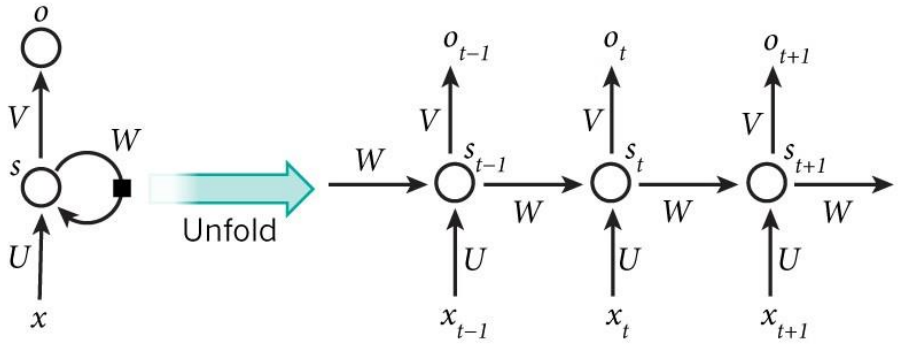


Figure 19. Recurrent neural network and unfolding.

LSTM networks are a special kind of recurrent neural networks, which can learn long-term dependencies from sequences. LSTM was first proposed by Hochreiter and Schmidhuber et al. [99] in 1997 and improved by Gers et al. [100] in 2000. LSTM is now very popular in many problems especially natural language text compression. LSTM is often implemented in "blocks" containing several units. This design is typical of deep neural networks and facilitates implementations with parallel hardware. In the equations below, each variable in lowercase italics represents a vector with a length equal to the number of LSTM units in the block. LSTM blocks control information flow via gates. Information can be stored in, written to, or read from a cell. The cell learns to make decisions about what to store, and when to allow reads, writes, and erasures, by opening or closing gates. The gates are implemented using the logistic function to compute a value between 0 and 1. Multiplication is applied with this value to partially allow or deny information to flow into or out of the memory. For example, an "input" gate controls the extent to which a new value flows into the memory. A "forget" gate controls the extent to which a value remains in memory. An "output" gate controls the extent to which the value

in memory is used to compute the output activation of the block. An architecture of LSTM is shown in Figure 5.

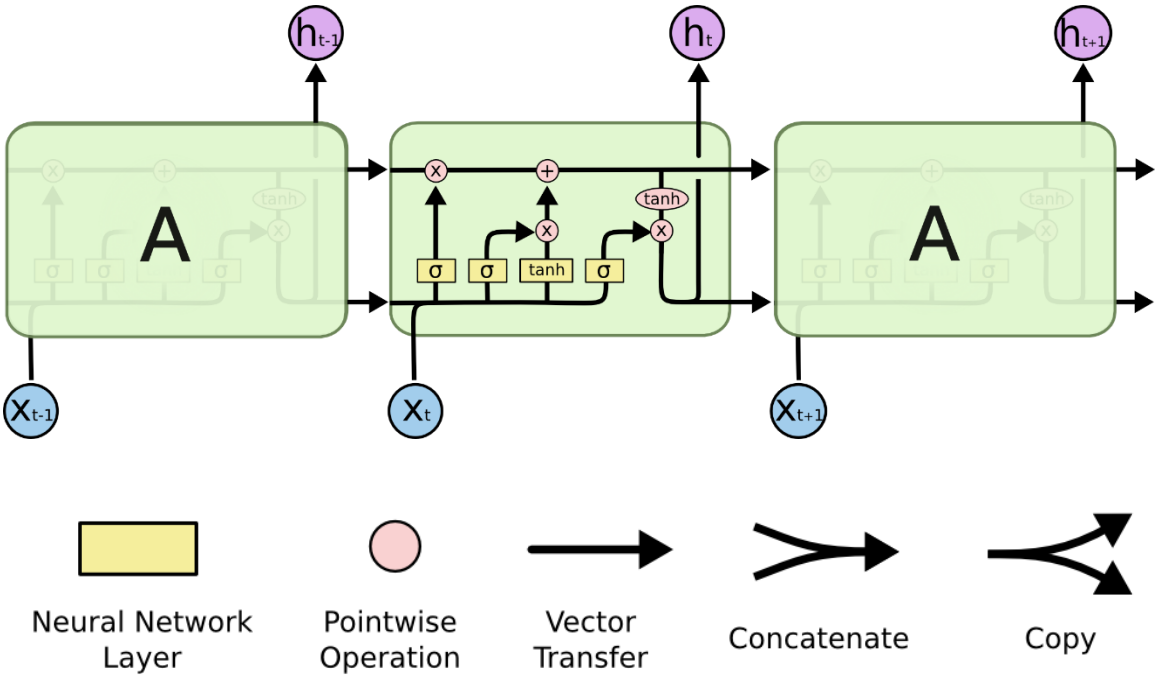


Figure 20. LSTM architecture.

The cell state updating equations are as follows, and Table 7 shows the notations:

$$f_t = \sigma_g(W_f x_t + U_f h_t + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot (W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \odot \sigma_h(c_t)$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\odot$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step. The weights in an LSTM block U and W are used to direct the operation of the gates. These weights occur

between the values that feed into the block (including the input vector x_t and the output from the previous time at step $h_{t-1}$and each of the gates. Thus, the LSTM block determines how to maintain its memory as a function of those values, and training its weights causes the block to learn the function that minimizes loss. To minimize LSTM's total error on a set of training sequences, iterative gradient descent such as backpropagation through time (BPTT) can be used to change each weight in proportion to its derivative with respect to the error. A major problem with gradient descent for standard RNNs is that error gradients vanish exponentially quickly with the size of the time lag between important events. With LSTM blocks, however, when error values are backpropagated from the output, the error becomes trapped in the block's memory. This is referred to as an "error carousel" that continuously feeds error back to each of the gates until they become trained to cut off the value. Thus, regular backpropagation is effective at training an LSTM block to remember values for a long term.

Table 7. Symbol list of LSTM model.

Symbol	Meaning
x_t	Input vector
h_t	Output
c_t	Cell state vector
W, U and b	Parameter matrices and vector
f_t	Forget gate vector. Weight of remembering old information.
i_t	Input gate vector. Weight of acquiring new information.
O_t	Output gate vector. Output candidate.
σ_g	sigmoid function
σ_c	hyperbolic tangent
σ_h	hyperbolic tangent

4.2.4 Convolutional Neural Networks

A Convolutional Neural Network (CNN) architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used, which will be reviewed below. For the readers to have a better understanding of how the layer works, we included some example figures as references. The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters

(or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

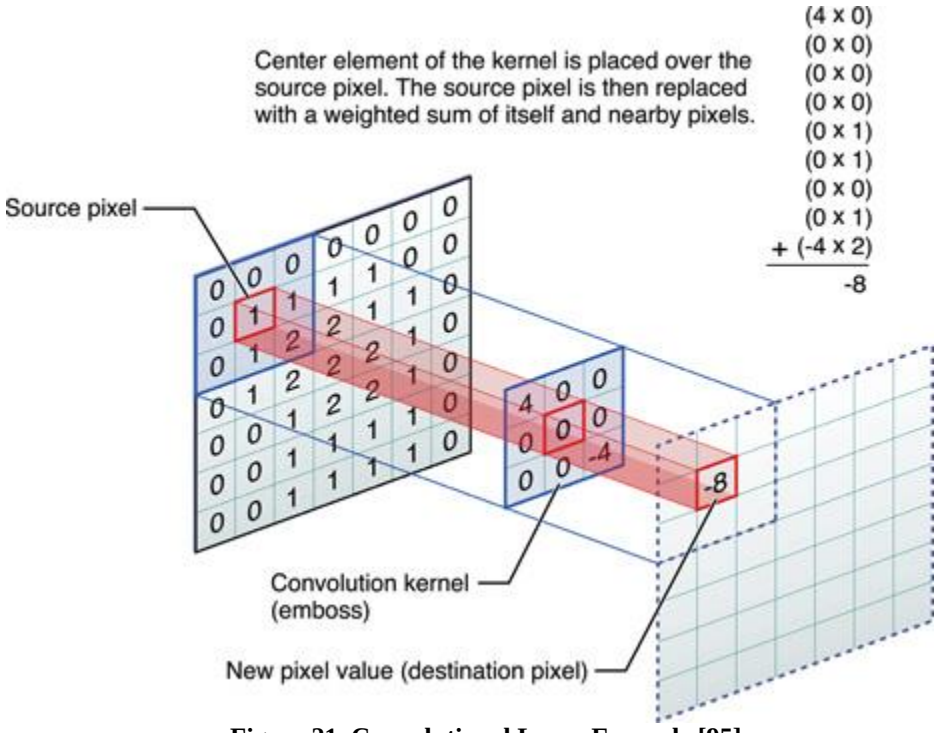


Figure 21. Convolutional Layer Example [95].

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping

rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance. The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged. Pooling is an important component of convolutional neural networks for object detection [96].

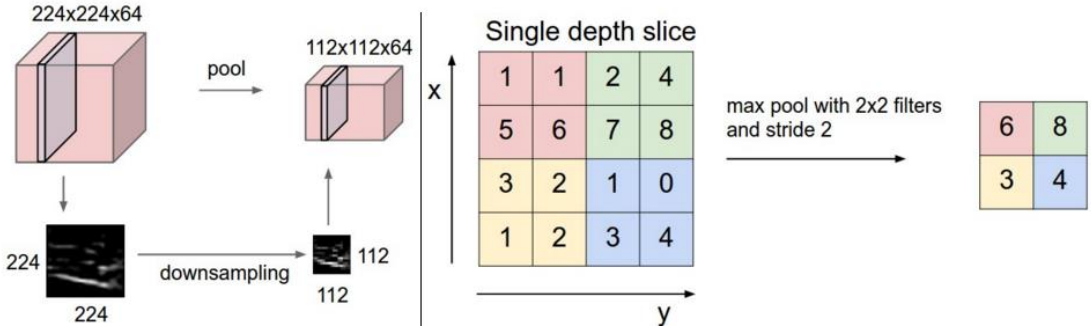


Figure 22. Max Pooling Layer Example [97].

Rectified Linear Units (ReLU) layer applies the non-saturating activation function $f(x) = \max(0, x)$. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other functions are also used to increase nonlinearity, for example the saturating hyperbolic

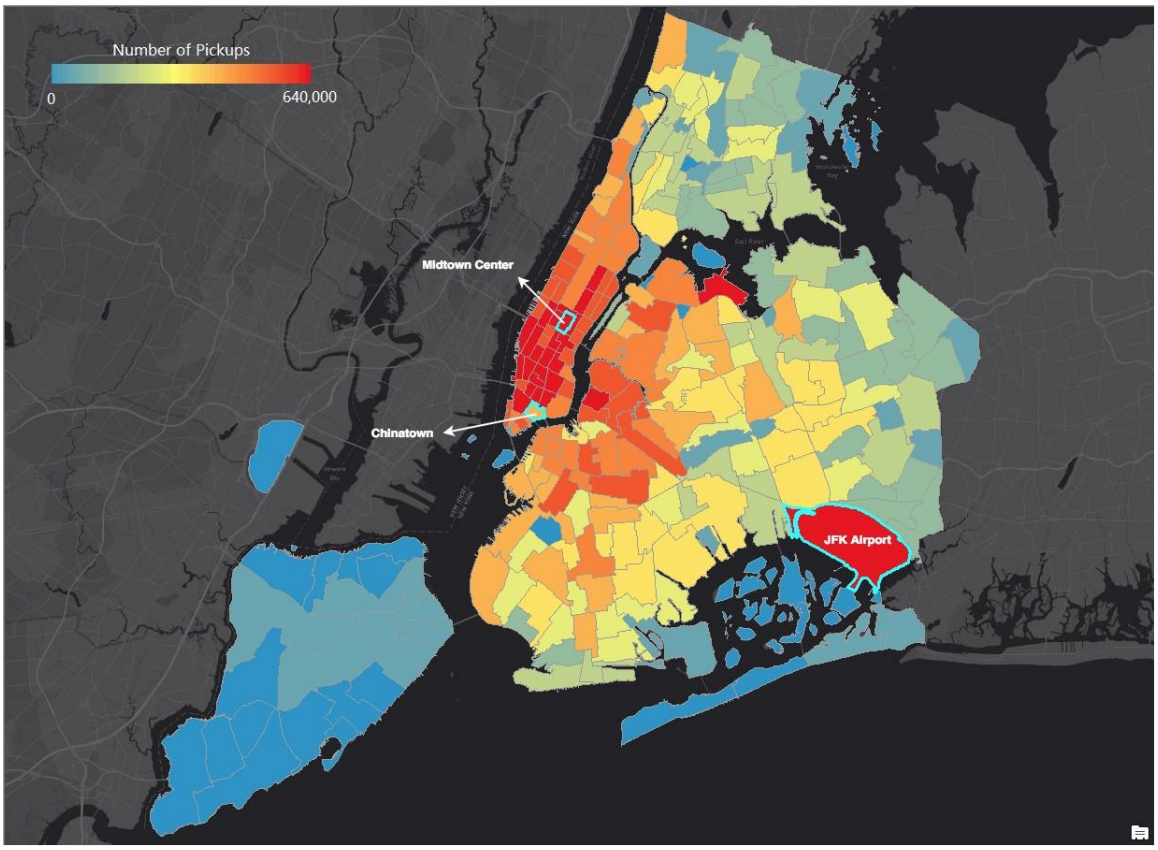
tangent $f(x) = \tanh(x)$, $f(x) = |\tanh(x)|$, and the sigmoid function $f(x) = (1 + e^{-x})^{-1}$. However, ReLU is often preferred to other functions because it trains the neural network several times faster [98] without a significant penalty to generalization accuracy. Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

4.3 Case Study: Predicting Real-time Uber Pickups in New York with LSTM

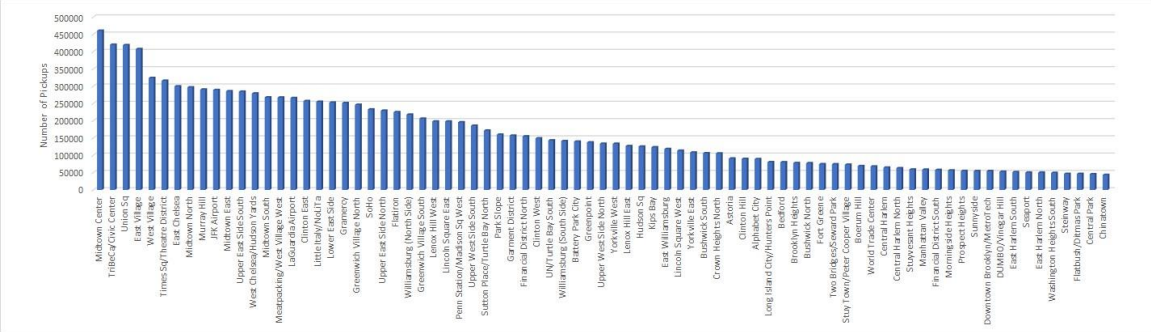
4.3.1 Data

The Uber pickup data used for this study open sourced by Fivethirtyeight [101]. This dataset contains data on over 4.5 million Uber pickups in New York City from April to September 2014, and 14.3 million more Uber pickups from January to June 2015. The data includes the pickup time and location (longitude and latitude for 2014 data, taxi-zone number for 2015 data) of each Uber trip. We used the first six months data in 2015 for this study. Figure 6 (a) visualizes the number of pickups in the 263 taxi-zones of New York City over the six months. It shows that Manhattan and the two airports (JFK and LaGuardia) have the densest Uber pickups. Figure 6 (b) shows the top 30% zones ranked by number of pickups. These taxi-zones contains 90% of the pickups. To evaluate the

proposed deep learning method, we chose 3 representative zones for analyzing: zone 161 (Midtown City), which is the is the "Uber heaviest" zone and contains the most number of pickups; zone 132 (JFK Airport), which reflects the airport ride demand; and zone 45 (Chinatown), which represent a less busy (but not too few of pickups to be trivial) pattern.



(a) Pickups heat map.



(b) Pickups for each census tract.

Figure 23. Uber pickups from January to June 2015.

4.3.2 Evaluation Metrics

To evaluate the accuracy of the prediction result, two well-known error measurements are used: the Symmetric Mean Absolute Percentage Error (SMAPE) and the Root Mean Square Error (RMSE). Consider $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n\}$ to be the Uber pickup predictions in time slot $[1, t]$, and $Y = \{y_1, y_2, \dots, y_n\}$ to be the true value of the pickup numbers. The RMSE and SMAPE are defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

$$SMAPE = \frac{100\%}{n}\sum_{t=1}^{n}\frac{2|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)}$$

4.3.3 Results

The prediction accuracy results for the three zones of interest are shown in Figure 24. For all 3 taxi zones, our proposed method outperforms the baseline ones and shows significant improvements. This means that the long-term dependencies captured by LSTM indeed have a positive influence on the prediction. Since the symmetric mean absolute

percentage error (SMAPE) is more sensitive to smaller observations and the root means square error, the variation of the prediction error for different taxi zones are reasonable. Figure 24 visualizes the prediction of the proposed method for the 3 zones of interest. Our deep learning model tracks the true value very well.

4.3.4 Conclusions and Future work

This chapter has presented a deep learning-based method to predict the number of Uber pickups. A long short-term memory (LSTM) model is proposed aiming to capture the long-term dependencies of the pickup sequence over time. Backpropagation through time (BPTT) is used to train our model. To evaluate the performance of the method, we applied it to predict the Uber pickups in 3 representative taxi-zones of New York City along with two baseline models: the time-varying Poisson model and the regression tree model. The results show that the proposed method has significant improvement to the baselines. For future work, it would be interesting to compare the Uber pickup data with New York taxi pickup data. Reference [92] shows some statistic temporal and spatial difference of Uber and taxi pickups. More advanced data mining techniques like tensor mining may help to discover deeper knowledge that may help to improve the Uber and/or taxi service system. On the other hand, with an accurate prediction of the pickups, we are able to design an Uber dispatching system and guide the Uber cabs to the area with high possibilities of pick demand. The efficiency improvement of the dispatching system would be a good topic.

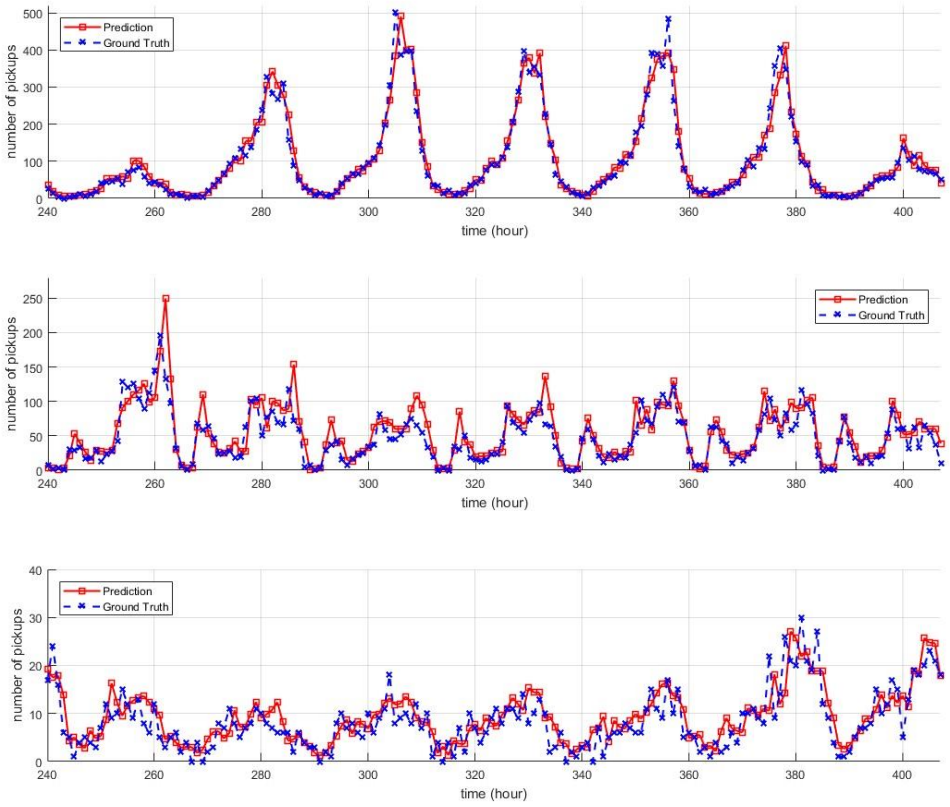


Figure 24. Prediction result by LSRM (10th week of 2015).

4.4 Case Study: TNC Real-time City-wise Demand Prediction with CNN and Context Information

4.4.1 Data

The data used in this study are the ride-hailing trip request data in Chengdu, China offered by DiDi Chuxing [102]. DiDi Chuxing is the world's largest and most valuable ride-sharing behemoth, with a monopolistic investment and merger and acquisition portfolio in the ride and bike sharing industry across the globe [103]. As a leading mobile transportation platform, DiDi receives more than 25 million trip orders, collects more than

70 TB of new route data, processes more than 4,500 TB of data, and obtains more than 20 billion queries for route planning and 15 billion queries for geolocation. In 2017, DiDi announced its GAIA Initiative and decided to share a complete sample of the route and ride request data with the academic community. The dataset was collected from November 1 to November 30, 2016, at Chengdu, China. The request dataset contains 7 million ride request records with origin-destination (OD) points and the trip start and end times. The route dataset offers 1 billion data points of the trip global positioning system trajectories with a sampling rate of 2–4s. The rich, dense records enable us to pursue high-resolution demand prediction at a local level. The dataset mainly used for this study is the ride request records. The information includes the pick-up and drop-off location and time for each ride. Table 8 lists the details of the data description. Figure 25 visualizes all the pick-up locations, which indicate the ride-hailing demand, over the observed month on a map of Chengdu.

The weather information is important for demand prediction considering the large impact of weather condition on the ride demand. We collected the open source hourly weather data from World Weather Online [104], including temperature, humidity, and weather conditions. Table 8 lists the parameters that we used in this study. For the convenience of model training, we assign each weather condition a numerical code.

Table 8. Ride request data from DiDi.

Parameter	Sample	Comment
Order id	mjiwdgkqmonDFvCk3nt Bpron5mwfrqvI	Anonymized
Trip Start time	1501581031	Unix timestamp, in seconds
Trip end time	1501582195	Unix timestamp, in seconds
Pick-up longitude	104.11225	GCJ-02 Coordinate System
Pick-up latitude	30.66703	GCJ-02 Coordinate System
Drop-off longitude	104.07403	GCJ-02 Coordinate System
Drop-off latitude	30.6863	GCJ-02 Coordinate System

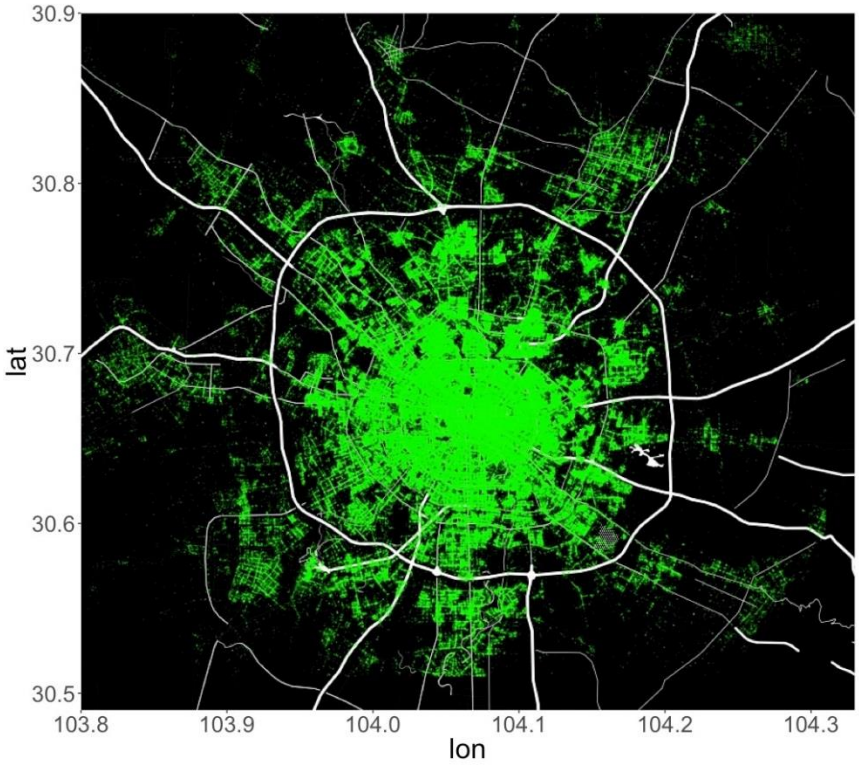


Figure 25. Visualization of the pick-up locations in Chengdu, China.

Table 9. Weather data parameters and weather condition encoding.

Parameter	Unit/Code	
Temperature	F°	
Humidity	%	
Weather condition	Unknown	0
	Clear	1
	Fog	2
	Haze	3
	Light Rain	4
	Light Rain Showers	5
	Mist	6
	Mostly Cloudy	7
	Partly Cloudy	8
	Patches of Fog	9
	Scattered Clouds	10

4.4.2 Problem Formulation

In the Uber pickup demand prediction, we were able to predict the next 1-hour total number of demands for a certain region in the city using an LSTM network trained by previous demand in that region. The goal of this study is to improve the capability of the prediction method in terms of accuracy, computational efficiency, temporal granularity, and spatial scalability. Since the DiDi dataset offers denser ride request data, we can divide the city region into smaller zones for demand prediction while still having enough data points at each region for model training. As shown in Figure 26, we select the urban core of the city and split it into 10 × 10 square grid cells. The longitude and latitude boundaries of the study region are also labeled. The high density of the ride-hailing demand also allows us to segment the prediction time interval in a smaller size and make more in-time prediction (e.g., predict the ride-hailing demand in the next 10 minutes). Another improvement we want to make is to involve the context information to the learning model,

which will potentially increase the accuracy of the prediction. For instance, weather conditions heavily impact the ride-hailing demands, e.g., there might be more rides when it's raining or snowing. Predicting pick-up demand further into the future is always desirable. Therefore, instead of only predicting the pick-up demand in the next time step, the proposed model can predict the demand for multiple time intervals in the future.

Let $D_t = [d_{1_t}, d_{2_t}, \dots d_{100_t}]$ be the number of demands for all 100 cells in time slot t, let $C_t = [min_t, day_t, t, temp_t, humi_t, wc_t]$ be the context-aware information of time t, where $temp_t, humi_t, wc_t$ represent the temperature, humidity, and weather conditions of time t, and min_t, day_t, t are the minutes of the day, day of week, and global time step. Based on the above considerations, we defined the problem that needs to be solved as follows:

> ***Given*** $[D_{t-m}, D_{t-m+1}, \dots, D_{t-1}]$ and $[C_{t-m}, C_{t-m+1}, \dots, C_{t-1}]$ as input,
> ***provide*** a prediction model that outputs D_{t+k},

where $m \geq 1$ is the number of past time steps used for prediction, $k \geq 0$ indicates the time step of ride-hailing demand that is predicted. In this study, m is set to be 6 and $k \in [0,5]$, and the duration for each time slot is 10 minutes. Therefore, the past 60-minute ride-hailing demands and weather records are used to predict the next 10–60 minutes of ride-hailing demand.

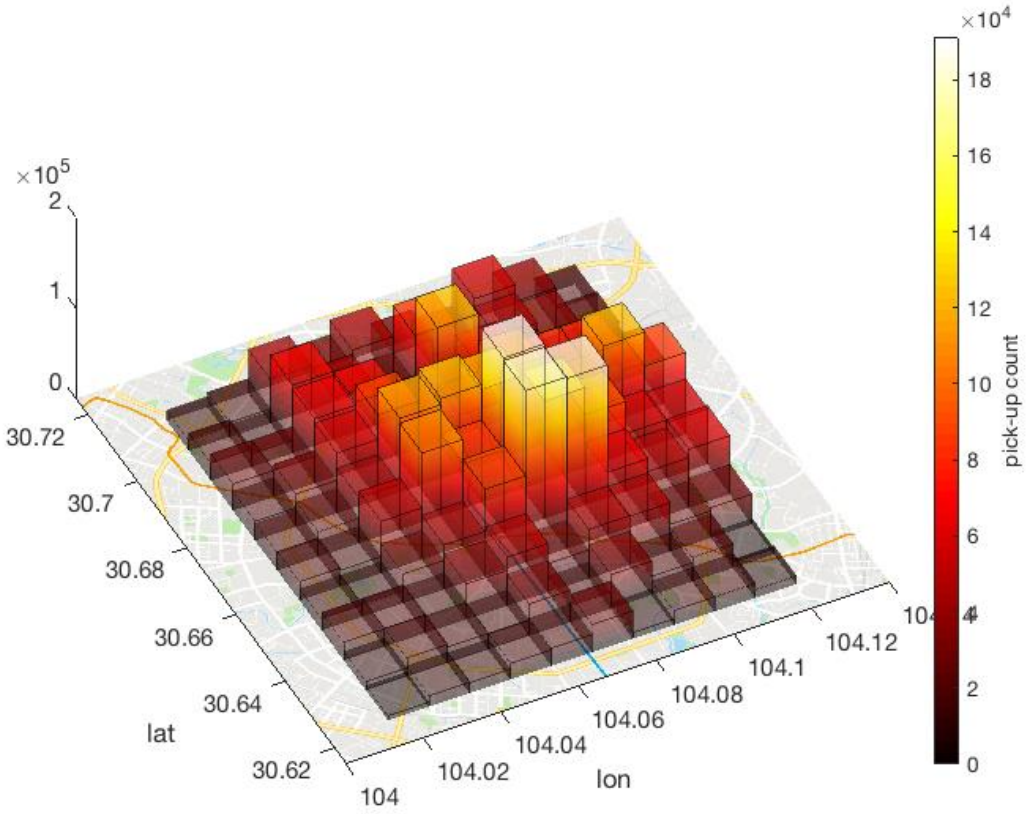


Figure 26. Ride-hailing demand distribution at Chengdu.

4.4.3 Methodology

In this section, the methods to develop the context-aware multi-step ride-hailing demand prediction model are presented. Two deep learning methods are designed and tested for this study, the LSTM and the CNN. The LSTM methods are developed in author's previous work [105] as a comparison. CNN is the major method that we focus on and explore in this chapter.

Referring to the problem description section, the input of the model is $[D_{t-m}, D_{t-m+1}, \dots, D_{t-1}]$ and $[C_{t-m}, C_{t-m+1}, \dots, C_{t-1}]$, where D_t is a 10 × 10 matrix representing the ride-hailing demand at time t for all 100 zones. Inspired by computer vision and image processing techniques, D_t can be treated as the frame of the image with 10 by 10 pixels. In order to involve the context-aware parameters, we added one more row of pixels to the frame and filled the first six pixels with the context-aware parameters $C_t = [min_t, day_t, t, temp_t, humi_t, wc_t]$. Thus, an 11 × 10 image input for the CNN is defined as shown in Figure 27. The "look-back" parameter is $m = 6$, meaning six frames are input to the network at one time. So the input we constructed is a tensor, and its size is 6 × 11 × 10, as shown in Figure 28.

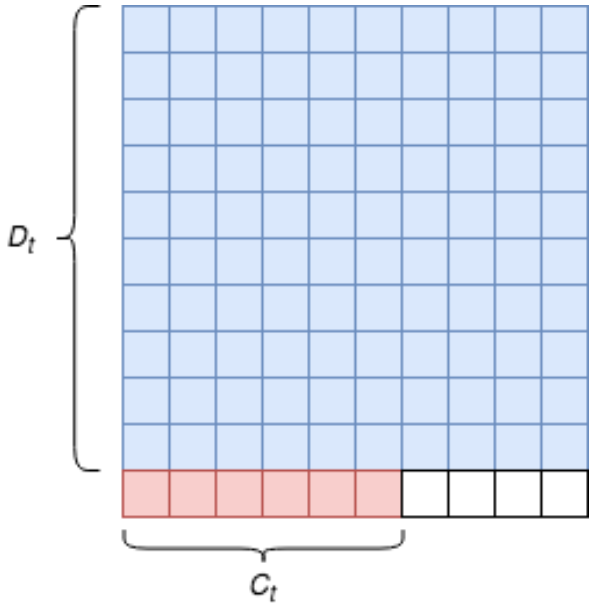


Figure 27. CNN input construction.

The designed CNN structure for this study is shown in Figure 28. The kernel size for the convolution and the max-pooling layer are labeled in the graph. The input is six frames of the matrices as we constructed above, representing that the model looks at the

demand of the previous six time steps (60 min) and predicts the future time steps. There are two convolution layers, each followed by a max-pooling layer for down sampling. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture [106]. The pooling operation provides another form of translation invariance, operates independently on every depth slice of the input, and resizes it spatially [107]. The most common form of the pooling layer contains filters of size 2 × 2 with a stride of 2. It downsamples at every depth slice in the input by 2 along both the width and height, discarding 75% of the activations. The depth dimension remains unchanged. To increase the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer, rectifier is used as activation function for the convolutional layer, also known as rectified linear unit (ReLU). The rectifier function gives an output x if x is positive and 0 otherwise:

$$f(x) = \max(0, x) \tag{1}$$

Following the two convolutional-pooling layer pairs is a flattening layer, which flattens the output from the last layer to a vector. The last two layers in the network are fully connected layers (256 hidden units) and the output layer. Since there are 10 × 10 regions in total for demand prediction, the output of the whole CNN is a 1 × 100 vector. For the multi-step prediction, we do not change the main characteristic of the CNN but only change the size of the output layer to output a $1 \times 100t$ vector, where t is the number of steps predicted ahead. We applied zero padding to pad the input volume with zeros

around the border while calculating the convolution to keep the output having same size with the input of the previous layer.

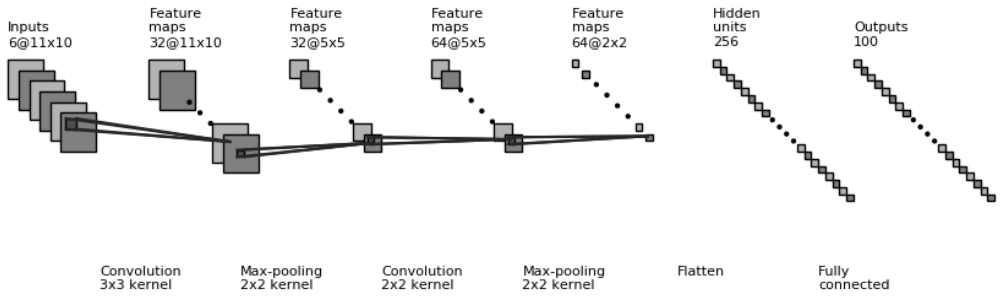


Figure 28. CNN architecture design.

4.4.4 Results

In this section, we present the results and findings of the proposed CNN model. In addition to CNN, two other models are used for comparison. One is a trivial instanton model that uses the current time slot-observed ride-hailing demand as the prediction of the next time slot, another is a LSTM based deep learning model. The main design of the LSTM network for this study is similar to the author's previous work [7, 21] with a few modifications. Instead of using the one-dimensional inputs and outputs in the previous work, multi-dimensional inputs and outputs are used to enable including the context-aware information in the model. The modified model structure allows the LSTM network to predict the ride-hailing demand of all zones simultaneously. Figure 29 visualizes the observation vs. the prediction of the three models. It shows that the predictions from the LSTM and CNN models are far better than the instanton model. For both LSTM and CNN, the correlation of the predicted demand and the observed demand is close to the ideal

condition represented by the red line. More details of the prediction error are presented in Table 10. The error metrics that we used are weighted mean absolute percentage error (WMAPE) and mean absolute error (MAE). They are defined as follows:

$$WMAPE = 100\% \cdot \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|} \tag{2}$$

$$MAE = \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3}$$

where y_i is the observation, $\hat{y}_i$ is the prediction, and n is the number of samples. The WMAPE is a variant of mean absolute percentage error (MAPE). It is designed for measuring the percentage error but avoids problems where a series of small or zero denominators are present. Table 10 shows that CNN performs slightly better than LSTM in terms of error measures WMAPE and MAE.

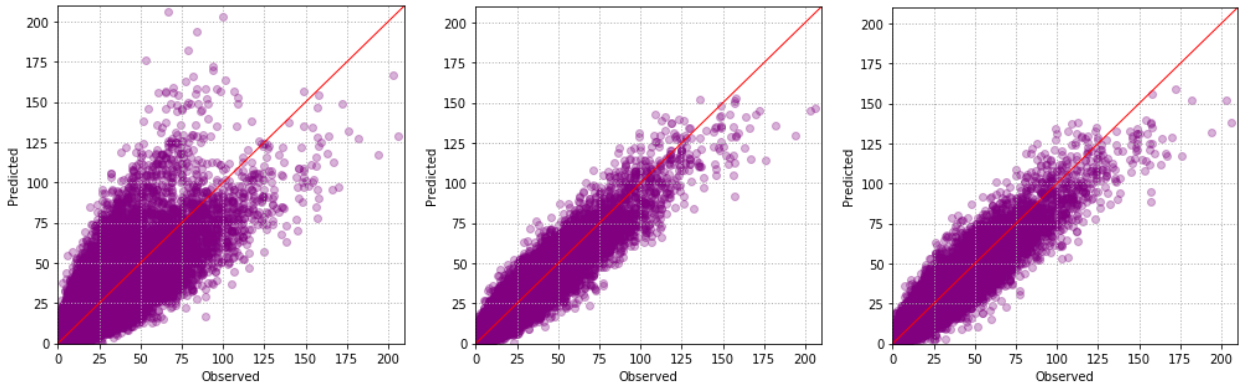


Figure 29. Observation vs. prediction for next 10-minute time step (from left to right: instantons, LSTM, CNN)

Table 10. Model prediction error comparison (for next 10-minute time step).

Model	Prediction Error	
	WMAPE	MAE
Instanton	46.84 %	4.6783
LSTM	24.97 %	3.1078
CNN	23.59 %	3.0616

The computational efficiency of LSTM and CNN is further examined. Table 11 lists the computing time for training and prediction for the two models. The comparison shows that despite more trainable parameters in the CNN model, the efficiency of the CNN model is over 30% faster for both the training and predicting when compared with LSTM.

Table 11. Model efficiency comparison.

Model	Training Time	Prediction Time	# Net Parameters
LSTM	17.59s	0.2057s	133,220
CNN	11.73s	0.1407s	172,452
Improvement	33.31%	31.60%	

For ride-hailing services, fleet operators need time to reassign vehicles to meet the future trip demand. Providing trip demand prediction further into the future is more useful than providing prediction of the immediate next time interval. Therefore, the accuracy of multi-step ride-hailing demand prediction was examined to test the long-term prediction capability of the proposed CNN model. The CNN architecture is slightly modified to accommodate 10, 20, 30, 40, 50, and 60 minutes ahead demand predictions. We only need to change the output size to fit the number of demands that are predicted for multiple steps, which is 100×6. Figure 30 indicates that as the prediction becomes further ahead in the future, the prediction error of the CNN model is increase very slowly within an acceptable level. This indicates that the proposed CNN model performs well in long-term prediction. Figure 31 shows the 60-minute ahead predictions vs. observations along time. It can be observed that the predictions align closely with the observed values.

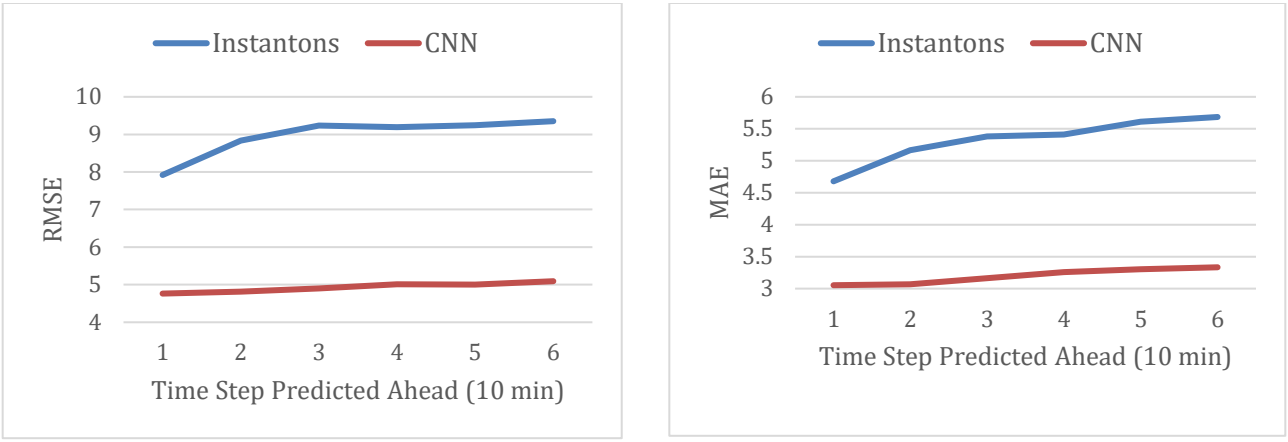
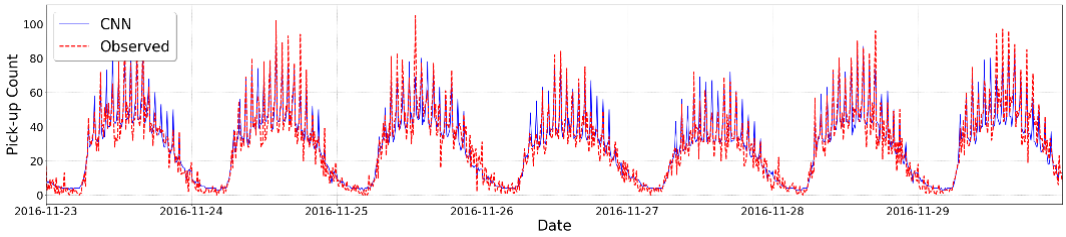
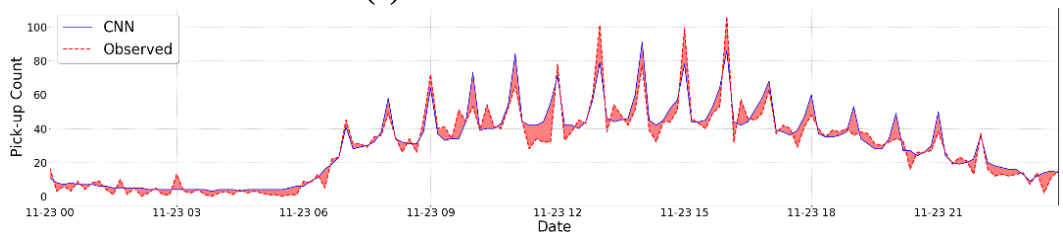


Figure 30. Error attenuation with multi-step prediction.



(a) 2016-11-23 to 2016-11-29



(b) 2016-11-23

Figure 31. 60-minute-ahead ride-hailing demand predictions vs. observation (Zone #56, which is represented by the pixel at row 6, column 6, refer to Figure 27).

4.4.5 Conclusions and Future Work

In this study, a CNN-based deep learning model is proposed for context-aware multi-step ride-hailing demand prediction. We utilized the 7-million trip records collected in Chengdu, China, provided by DiDi Chuxing to train and test the model. The outcomes and findings are promising. We split Chengdu City into 10 × 10 square zones with 1-km

side lengths. The CNN model can provide accurate demand predictions for all 100 zones every 10 minutes. The prediction accuracy significantly outperforms the baseline model and produces slightly lower error measures than the LSTM model, which the authors proved to effectively model ride demand in earlier published works. The computational efficiency of the CNN model is further examined. The result shows that although the number of trainable parameters in the CNN model are higher than in the LSTM model, the CNN model is 30% more computationally efficient for both training and predicting. The proposed model can also be easily extended for multi-step predictions that could be applied for on-demand shared automated vehicle operations. We found that the CNN model prediction accuracy is still satisfied when predicting ride-hailing demand 60 minutes ahead.

Predicting ride-hailing demand can benefit ride-hailing vehicle operation efficiency. For future work, a ride-hailing fleet dispatching system will be developed based on the demand prediction. Prior knowledge of the demand distribution around the city would help operators dispatching vehicles to the passengers' nearby locations before they make a ride request to provide more in-time service. Proactive fleet management will save vacant time and travel distance for the vehicles between rides.

Chapter 5

Deep Learning based Realtime Computer Vision System for Lane Change Behavior Detection

5.1 Introduction

The last chapter discusses the knowledge discovery and data mining from macroscopic big data with a specific research focus on ride-hailing demand prediction with historical activity data. In this chapter, the study scope is switched to microscopic and the single vehicle data will be used for information inference. As the most rapidly developed technology, autonomous driving systems highly depend on environment understanding with large scale perception data. It is essential to develop advanced data mining technologies to improve the information extraction accuracy and speed. Most autonomous driving systems perceive their surroundings through a wide variety of sensors, such as radar, LiDAR, GPS, cameras and the inertial measurement unit (IMU) [108]. An internal map is then generated with collected inputs and guides the autonomous vehicle driving in a safe, fast and energy-saving manner [109]. Various techniques and applications have been developed in the fields of intelligent transportation systems to achieve different levels of vehicle automation, including speed assistance, collision avoidance, lane keeping, etc.

[110]. Among them, the vehicle localization technique is one of the most important because an accurate positioning system allows the autonomous vehicle to understand its surrounding with less effort and operate the driving command safely. The most common vehicle localization method is to use GPS, which provides absolute positioning information at a low cost. However, GPS is vulnerable to various interference, such as tall buildings, trees or other signals which could prevent the GPS device from receiving satellite signals [111]. Moreover, the positioning error could reach up to tens of meters and therefore only road level resolution can be achieved [112]. Therefore, other methods are invented to fuse with GPS so that they can compensate for the large positioning error or used as back-up sensors when GPS is temporarily unavailable. For example, GPS - LiDAR fusion technique uses the LiDAR point cloud to estimate the incremental motion and model the error covariance in LiDAR-based position measurements, and the globally referenced pose is calculated using Unscented Kalman Filter given the measured error [113]. IMU enables the dead-reckoning method, but its error accumulation can cause errors for long term use [114]. Cameras, which usually act as visual odometers, can capture more information and are relatively cost-effective. By analyzing the lane-changing behavior of the vehicle through camera images and integrating the obtained information with a GPS, the tracking algorithm could tell the exact lane the vehicle is on and the localization accuracy can be enhanced up to centimeter [115]. The inexpensive vision-based localization method has drawn tremendous attention of researchers and a series of lane-changing detection applications have come up throughout the years to improve the detection accuracy.

While many early efforts on camera-based lane departure detection rely on lane boundary modeling [116], [117], [118], the overall image information is not considered comprehensively, making the system very sensitive to camera calibration and lack of potential to adapt complex conditions like high-density traffic. In [119], the authors employed a support vector machine (SVM) based framework to detect the lane-changing behavior using the edge information extracted from the pre-defined region of interest in the original image. The principal component analysis (PCA) was applied to reduce the dimension of the image features while keeping its energy, and the algorithm reached an accuracy of 68.5% when tested on actual driving data. A convolutional neural network (CNN) [120] based lane-change classifier was also implemented using the extracted edges as input and reached an accuracy of 79.7%. Higher detection accuracy was reached in [121], where the author applied a stacked sparse autoencoder model to classify the lane changing behaviors using the extracted useful features from images. A series of image preprocessing techniques were used to remove noise and enhance the classification accuracy, such as graying, filtering, binarization and setting a dynamic region of interest. An accuracy of 96.69% was achieved when testing on a total of 5309 frames of image sequences. A linear model was used for lane detection, therefore the algorithm might not work well for a larger scale of implementation with more complex road situations. In recent years, deep learning (DL) approaches are increasingly applied to vision-based lane departure detection for end-to-end solutions. In [122], lane positions were estimated using an end-to-end deep neural network. Lanes were carefully marked in the images captured by a laterally-mounted down-facing camera and used as labels to the network. The detected

lanes could then be used for the lane changing detection or lane departure warning. Despite the various vision-based lane detection methods, there hasn't been an end-to-end lane-changing behavior detection system using captured images directly.

In this chapter, we proposed a vision-based real-time system to detect the lane-changing behavior of the vehicle. We designed deep residual learning neural networks to recognize the images captured by a forward-facing, in-vehicle camera and determine the three lane-changing behaviors: lane departure to the left (class 1); to the right (class 2); or no lane departure (class 3). To further improve the accuracy, another network was designed to utilize the IMU information as additional input. A baseline model using only IMU data was also developed for comparison. The NUDrive 1000 lane-change dataset [123] was used to train, validate and test the proposed models. The results indicate the proposed method achieves good lane change detection accuracy and is capable to work in real time. The main contribution of this chapter is to propose a deep learning based computer vision system for lane change behavior, characterized by these key novelties: 1) It offers a novel end-to-end solution directly from image to lane change detection and no other model (like lane marks detection model) in the middle is involved; 2) As one of the leading edge deep learning algorithms, to the authors' knowledge, ResNet has not been customized and integrated with lane change behavior detection yet; 3) We improve the performance of the image-input-only ResNet by novelty concatenating the IMU data with the feature map (the pooling layer) before the fully connected layer; 4) Our system effectively and efficiently detects the lane-changing behavior with 87% of accuracy and 0.028s of response time.

5.2 Methodology

5.2.1 Data Collection, Fusion, and Pre-processing

In this work, the NUDrive 1000 lane change dataset [123] is applied to study the lane-changing behavior. The driving data was collected on real highways around Nagoya, Japan. Video images of the road in front of the vehicle were recorded by a front-facing camera, and vehicle operation signals including vehicle velocity, acceleration, and gas and brake pedal pressures were recorded using different sensors. Ten drivers were recruited to complete approximately 50 km of highway driving for each driver. The images were sampled from the driving videos at 10Hz and labeled with one of the three lane-changing behaviors, lane departure to the left, to the right, or no lane departure. Table 12 shows the three class labels and how they are represented in the dataset. The beginning and end of lane changes are defined as the time when lane markers appeared to begin moving or stop moving laterally on the front-view video respectively. Sample images representing the three classes are shown in Figure 32 respectively. The dataset is large enough (around 200k training and 25k testing images) to prove the effectiveness of the algorithm when applied to real-world highway driving scenarios.

Table 12. Classes of Lane-changing Behavior.

Class	*Behavior*	*Class Representation*
1	Lane departure to the left.	-1
2	Lane departure to the right.	1
3	Keep in lane.	0

As shown in Figure 33, we crop out the upper and lower border of each image, where useful information is rarely contained. The smaller image has less noise and saves computational power for both training and testing. For each image, there is a set of corresponding IMU data which contains information of brake and gas pedal force, velocity, steering angle, and longitudinal and lateral acceleration of the vehicle. A detailed description of the IMU data is presented in Table 13. Those IMU data are combined as a 6-entry vector for the model input.

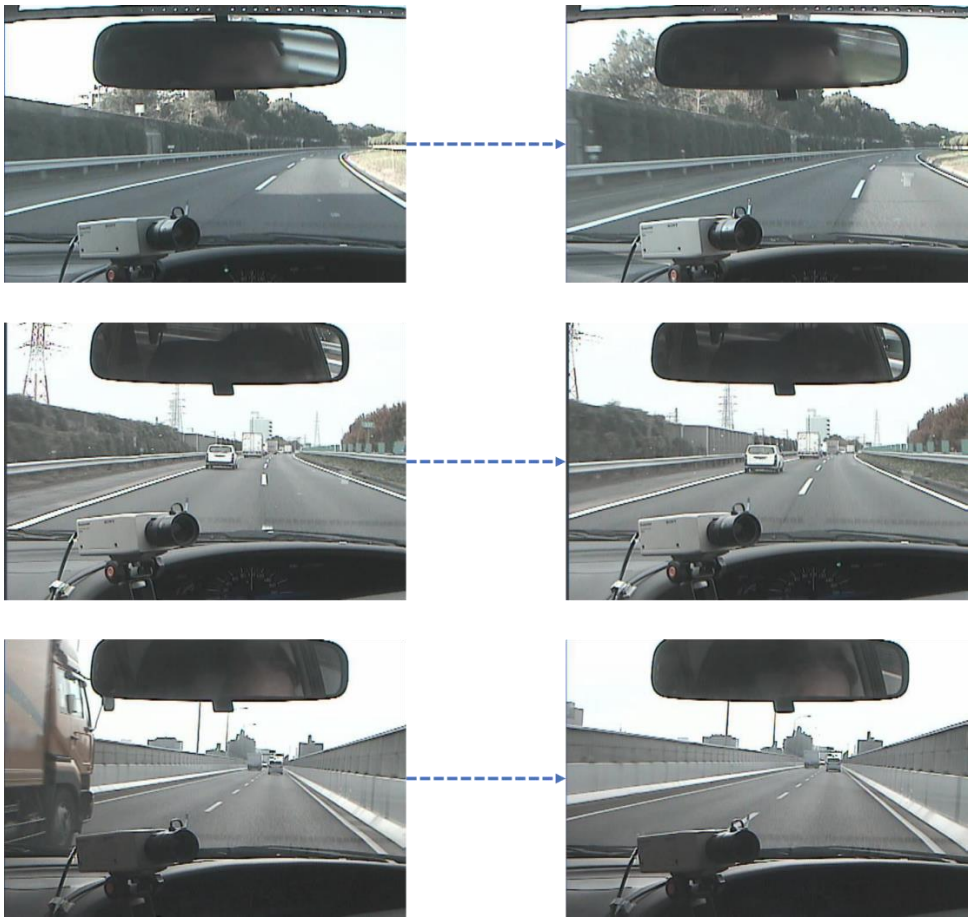


Figure 32. Sample images of the three lane-changing behaviors: lane departure to the left (upper); to the right (middle); or no lane departure (bottom).

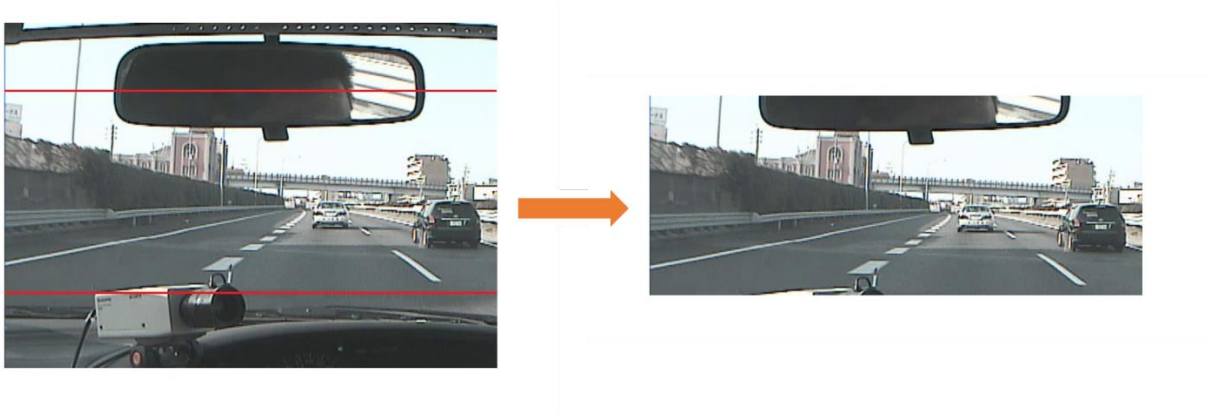


Figure 33. Image input is partially cropped to get rid of useless information. The cropped image has size 278×692 compared to the original image (480×692).

Table 13. Description of IMU Data.

Signal	*Description*
Force on brake pedal [N]	Force on the brake pedal obtained by a pressure sensor.
Force on gas pedal [N]	Force on the gas pedal obtained by a pressure sensor.
Velocity [km/h]	Vehicle velocity obtained by a pulse generator.
Steering angle [deg.]	Steering wheel angle obtained by a potentiometer.
Longitudinal acceleration [G]	Straight line acceleration obtained by an accelerometer.
Lateral acceleration [G]	Centripetal acceleration obtained by an accelerometer.

5.2.2 Problem Description

The real-time lane change detection aims to understand the driving behavior and determine the current lane-changing status, which can be modeled as a classification problem:

$$y = f(x) \qquad (1)$$

where f represents the classifier, $y \in \{-1, 0, 1\}$ is the output representing the class label, x represents the input observation. The main objective of this study is to train a

classifier that can take a frame of image (and/or IMU data) as input, and output the class of lane-changing state of the vehicle. As introduced above, we have two different types of observations: image and IMU. For each instance, the image is expressed by 3-channel 2D pixels and the IMU data is a 6-entry vector. To fully evaluate these two types of input, we develop separate models to test the detection accuracy for the 3 different input combinations, IMU only, image only, and image combined with IMU. Three models with different structures are trained to accommodate those inputs. The description of the method will be introduced in the following subsections.

5.2.3 Network Architecture for Image Input Only

For the model with image input only, a deep neural network is trained using an adapted design of ResNet [124] with its architecture shown in the flow chart in Figure 34. Resnet develops a residual learning framework which enables faster optimization with deeper network structure. This avoids a common problem named as “degradation” for “plain” net, in which simply stacked layers commonly result in higher training error when the network depth increases [125].This network allows us to train with more layers and larger datasets at a faster speed (~ 200k images trained in 80 hours in our case).

The network receives three feature maps (RGB channels) as input, followed by the first convolutional layer (Conv) which expands the number of channels to 16. After the initial expansion, 12 residual blocks are connected after one another which increases the number of channels to 1024 and downsamples the feature map to 1/32 of its original size. The residual blocks would help improve the training speed of the network with its structure shown below:

$$Y = X + \mathrm{Re}\,LU(Conv(\mathrm{Re}\,LU(Conv(X)))) \qquad (2)$$

where ReLU is the Rectified Linear Unit used as the nonlinear activation function after convolutional layers, defined as ReLU(Z) = max(0, Z). X is zero padded to match the increasing dimension of the convolutional layers. After the residual blocks, an average pooling layer that eliminates the first two dimensions is used so that images of any size can be feed into the network without dimension mismatch. Two fully connected layers (FC) and a softmax layer (Softmax) are used in the end that output a 3-entry vector as an indication of the probability that image belongs to each of the three classes. The softmax function is defined as:

$$y_i = \frac{e^{z_i}}{\sum_{k=1}^{3} e^{z_k}} \quad \text{for i = 1, 2, 3} \qquad (3)$$

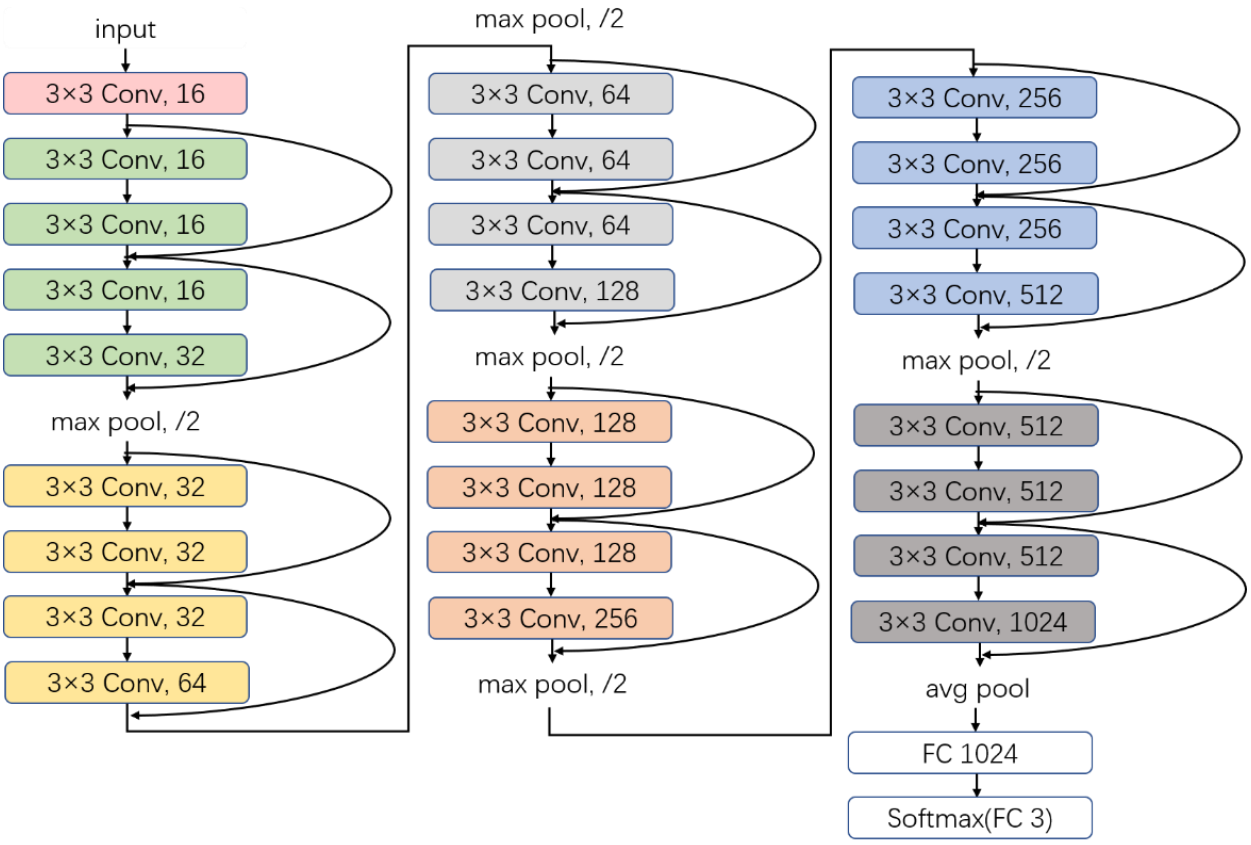


Figure 34. Network architecture trained for lane-changing detection using only images. The network contains 27 layers with around 1.1 million trainable parameters.

The label is set to be [1, 0, 0] for class 1, [0, 1, 0] for class 2 and [0, 0, 1] for class 3 and the network loss is defined as a cross entropy loss with equation shown below:

$$L = -\sum_i y_i \log x_i$$

$$Y = X + ReLU\left(Conv\left(ReLU\left(Conv(X)\right)\right)\right)$$

where y_i and x_i are the entry of label and network output respectively. The kernel size is set to be 3×3 with stride equals 1 for the convolutional layers and 2×2 with stride

equals 2 for the max-pooling layers. During the training, the learnable parameters including weights, biases, and filters are updated using an adaptive moment estimation (Adam) [126] optimizer with a learning rate of 1×10^{-4} and batch size of 16. A summary of the output size and dimension of the convolutional filters at each layer is shown in Table 14. Conv_1 is the first convolutional layer and Conv_*i*_*j* is the *j*th layer in the *i*th block. A block contains two residual connections and is represented by the same color in Fig 3. Max pooling layer that downsamples the image by 4 is performed at Conv_2_4, Conv_3_4, Conv_4_4, Conv_5_4, and Conv_6_4, and an average pooling layer is performed after Conv_7_4 to transform the output size to 1024×1.

Table 14. Network Outputs and Convolutional Kernels Sizes

Layer Name	Output Size	Filter dimension
Conv_1	278×692	[3×3, 16, stride 1]
Conv_2_x	278×692	[3×3, 16, stride 1] × 2
		[3×3, 16, stride 1] [3×3, 32, stride 1]
Conv_3_x	139×346	[3×3, 32, stride 1] × 2
		[3×3, 32, stride 1] [3×3, 64, stride 1]
Conv_4_x	70×173	[3×3, 64, stride 1] × 2
		[3×3, 64, stride 1] [3×3, 128, stride 1]
Conv_5_x	35×87	[3×3, 128, stride 1] × 2
		[3×3, 128, stride 1] [3×3, 256, stride 1]
Conv_6_x	18×44	[3×3, 256, stride 1] × 2
		[3×3, 256, stride 1] [3×3, 512, stride 1]
Conv_7_x	9×22	[3×3, 512, stride 1] × 2
		[3×3, 512, stride 1] [3×3, 1024, stride 1]
FC_1	1024×1	
Softmax (FC_2)	3×1	

The network is implemented using TensorFlow [127]. The training process takes ~400,000 iterations, which is about 34 epochs. All the model training, validation, and testing were all performed on a PC with four-core 4.20 GHz CPU, 64GB of RAM, and Nvidia GeForce GTX 1080 GPU.

5.2.4 Network Architecture for Image and IMU Combined Input

We modified the previous network structure so that it could receive both types of input. We concatenated the one-dimensional IMU data with the average pooling layer in Figure 35 before feeding them into the fully connected layer, as shown in Figure. 35, and kept all the other layers the same. The omitted structure before the blue arrow is the same as the layout of the convolutional layers in Figure 34, Also, the exact training parameters including learning rate, batch size, and loss function are set to be the same. This is to ensure that the structure and number of trainable parameters are close to the previous network and a fair comparison can be made to the detection accuracy with and without the additional IMU input.

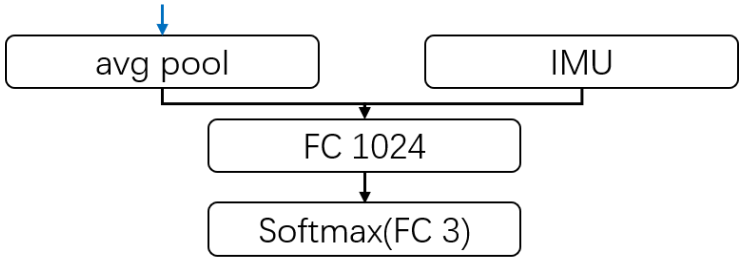


Figure 35. Concatenated IMU data with an average pooling layer.

5.2.5 Model for IMU Input Only

Using only IMU data to make lane change behavior predictions were well studied in the first Intelligent Transportation Systems plus Data Mining challenge during 2017 IEEE Intelligent Transportation Systems Conference [128]. There were twenty-three academy and industry contestants, coming from eleven different countries competed in the challenge and offered a variety of statistical and machine learning solutions. Among the

solutions, the gradient boosting tree and the random forest are the top two models ranking with prediction accuracy. We reproduced the gradient boosting tree model as the baseline for the proposed vision-based methods.

Tree boosting is a highly effective and widely used machine learning method. Usually, a single tree is not strong enough to be used in practice. Tree boosting is an ensemble model which sums the prediction of multiple trees together. The method we used is a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. More details of this method can be found in [129].

5.3 Experiment Setup, Result and Analysis

All the three models are trained with the same dataset which contains 187,440 training images, 4500 validation images, and 24,626 testing images with each image of size 480 × 692. And for each image, there is a corresponding IMU containing 6-entry vector. The training input is fed into the model to train the parameters, the best model is chosen from the least loss or highest accuracy validated using the validation dataset, and the obtained model is tested with the testing dataset to calculate the accuracy. All the testing images are in different video clips from training and validation dataset to ensure the trained model is applicable to different road scenarios.

5.3.1 Testing Result for Training with IMU Only

In [128], the gradient boosting tree achieves 86.9% testing accuracy. However, considering the highly unbalanced testing data with 83.8% non-lane change rate, the accuracy is only 3% above a trivial guess.

In order to compare the baseline with proposed methods under the same condition, we applied the reproduced method to the same training, validation and testing set used for our proposed methods. Due to a 51.6% of non-lane change rate in our testing set, the testing accuracy is 55.6%, which is 4% above a trivial guess, indicating a similar performance as it was realized in [128]. A detailed test result is shown in Table 15.

Table 15. Testing Result for Tree Boosting Model Trained with IMU Data Only

Result	*Class 1*	*Class 2*	*Class 3*	*Total*
Training Data	18778	19178	149484	187440
Validation Data	1500	1500	1500	4500
Testing Data	5898	6033	12695	24626
Testing Positive	587	880	12216	13683
Testing Negative	5311	5153	479	10943
Testing Accuracy	9.95%	14.59%	96.23%	55.56%

5.3.2 Testing Result for Training with Images Only

After the training of a deep CNN with structure from Figure 34, the testing results are listed in Table 16. The results show that the proposed network structure is capable to identify the three categories of lane-changing behavior at an accuracy of 85.43%. The convergence plot is shown in Figure 35. The training accuracy achieves 90% at around 35000 iterations.

Table 16. Testing Result for Training with Image Data Only.

Result	*Class 1*	*Class 2*	*Class 3*	*Total*
Testing Data	5898	6033	12695	24626
Testing Positive	5304	5093	10640	21037
Testing Negative	594	940	2055	3589
Testing Accuracy	89.93%	84.42%	83.81%	85.43%

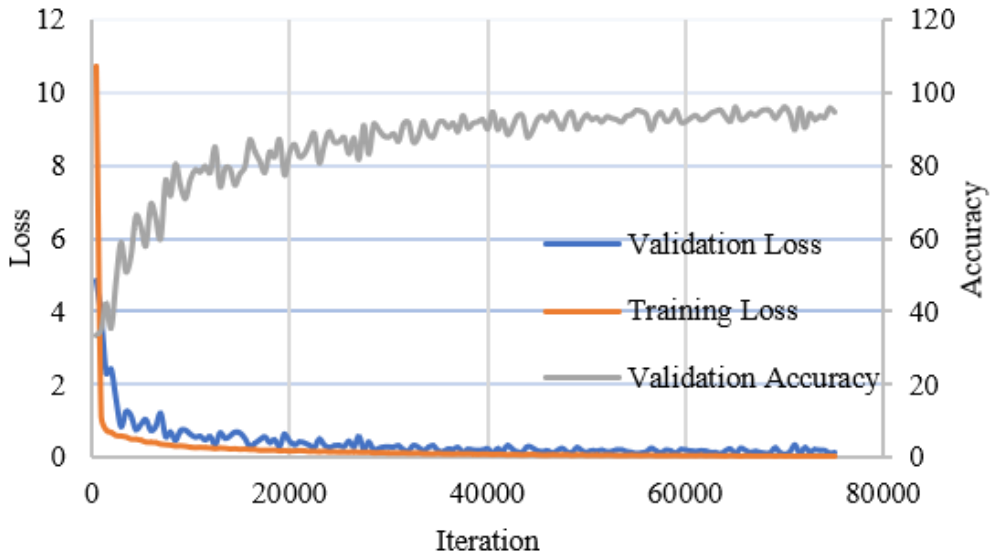


Figure 36. Convergence plot for the network trained with image only. The training takes a total of ~410k iterations, and 75k iterations are shown here since the convergence is too slow after 60k iterations.

5.3.3 Testing result for Training with Image and IMU

After training of neural network with image and IMU combined CNN structure, the testing results are listed in Table 17. The proposed network outperformed the previous two networks with an accuracy of 86.95% showing the network capability of utilizing two types

of data. The convergence plot is shown in Figure 37. The training accuracy reached 90% at around 35000 iterations.

Table 17. Testing Result for Training with Both Image and IMU Data.

Result	*Class 1*	*Class 2*	*Class 3*	*Total*
Testing Data	5898	6033	12695	24626
Testing Positive	5194	5350	10868	21412
Testing Negative	704	683	1827	3214
Testing Accuracy	88.06%	88.68%	85.61%	86.95%

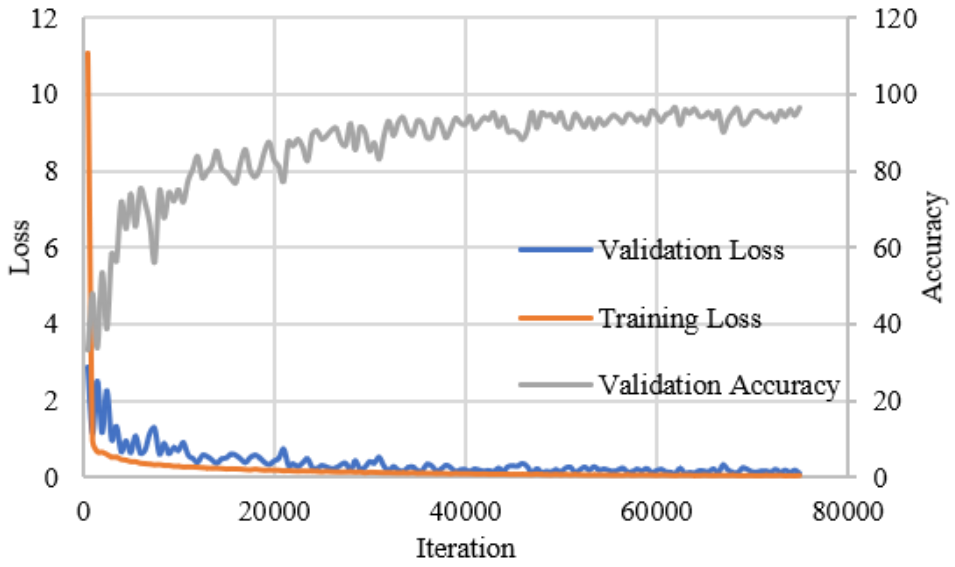


Figure 37. Convergence plot for the network trained with both image and IMU data.

5.3.4 Comparison and Discussion

Speed of convergence comparison between network trained with image data only and images combined with IMU is shown in Figure 37. The similar decreasing of losses shows that the two training processes have a similar speed of convergence. The training

time, the number of iterations, testing time and accuracy for each method are listed in Table 18.

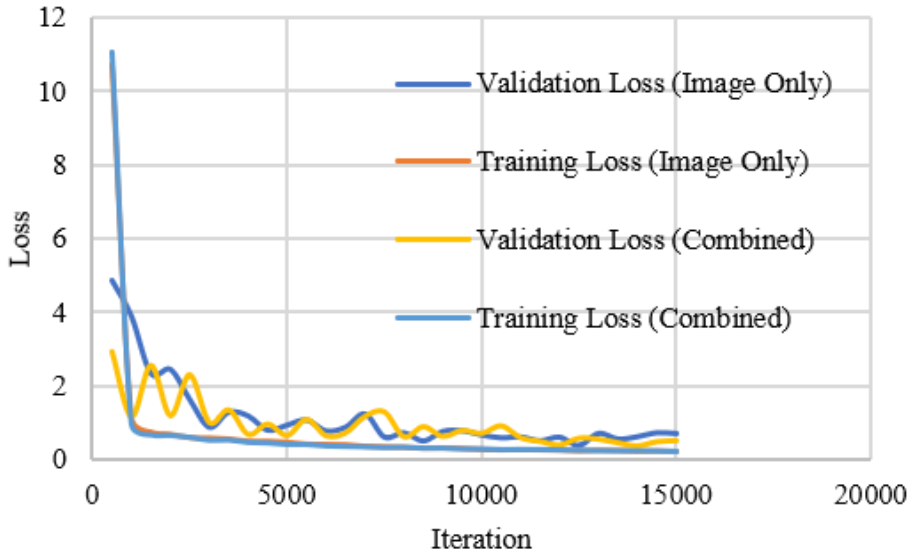


Figure 38. Speed of convergence comparison between the network trained with image data only and images combined with IMU. It shows that the two training processes have a similar speed of convergence.

Table 18. Training Time, Number of Iterations, Inference Time and Accuracy for Different Methods.

Model	*Training time (s)*	*Iteration*	*Inference Time (s/image)*	*Accuracy*
IMU Only	101	8500	3.44×10^{-7}	55.56%
Image Only	303733	412000	0.0276	85.43%
Image + IMU	519724	647500	0.0278	86.95%

The network trained with only IMU performs the worst because IMU data contains the least amount of information, only 6 values per time step. On the other hand, images contain way more information than IMU data and give a better result. The combination of

the two achieves the best result. The inference time for a single image using the proposed models is below 0.028 seconds, which is capable of processing over 35 tuples of input per second. This means the proposed models reach the real-time level of computational speed. For a camera with a frame rate of 60, the lane change detection is able to respond for less than every two frames, which is quick enough for making life-saving decisions. In comparison, the average reaction time for humans is 0.25 seconds [130] to a visual stimulus. Our models are 9 times faster than human reaction.

5.4 Conclusions and Discussion

This research explores end-to-end vision-based real-time detection approaches for identifying highway lane-changing behavior using deep learning. We designed deep residual learning neural networks to recognize the images captured by a forward-facing, in-vehicle camera and determine the three lane-changing behaviors: lane departure to the left (class 1); to the right (class 2); or no lane departure (class 3). To further improve the accuracy, another network was designed to utilize the IMU information as additional input. A baseline model using only IMU data was also developed for comparison. The NUDrive 1000 lane-change dataset was applied to train, validate and test the proposed models. The testing results on over 24k images show that the proposed method can achieve a detection accuracy of 55.6% on using only IMU data, 85.43% on using only image data and 86.95% on using both image and IMU data. The testing time of 0.0278 s/image also indicates the real-time working ability of the proposed method. Compared to the average human reaction to visual stimuli, the proposed computer vision system works 9 times faster, which makes

it capable of helping make life-saving decisions in time. In the future, more research will be conducted as listed below:

- Extend the highway lane changing identification system to a local street where road scenarios are more complex.
- Since the data is in time sequences, we would like to test different recurrent neural network (RNN) structures, ex: Long short-term memory (LSTM), where the entire sequence of data is analyzed, and compare with the current network result.

Chapter 6

A Mesoscopic Simulation-based Framework to Evaluate the System-Level Impact of Connected and Automated Vehicles Coupled with Shared Mobility

6.1 Introduction and Motivations

In Chapter 4 and Chapter 5, both macroscopic and microscopic cases were studied for knowledge discovery and data mining for shared mobility and CAV applications. To further understand and predict how emerging ride-hailing and CAV modes will affect the transportation system, mesoscopic agent-based simulations are necessary. With the rapid development of vehicle and communication technology, CAVs have emerged as a prospective solution to a number of transportation problems, as well as bring forth new mobility and energy efficiency opportunities. With wireless communications between vehicles and infrastructure or among vehicles, more efficient vehicular maneuvers have been achieved through information sharing and better cooperation, including Cooperative Adaptive Cruise Control (CACC) [131], Cooperative Lane Change [132], Cooperative Ramp Merging [133], and Signalized Intersection Eco-Approach/Departure (EAD) using

anticipated SPaT information [134]. In United States, cooperative automated driving studies and deployment focused on CACC and EAD, as they bring significant improvement on mobility and energy efficiency, even at scenarios when the automation level and penetration rate of CAVs are relatively low.

Taking advantage of vehicle-to-vehicle (V2V) communications, CACC allow vehicles to form platoons and be driven at harmonized speeds with shorter time headways between them [135][136]. By sharing vehicle information such as acceleration, speed, and position in a distributed manner, CAVs in a certain communication range can cooperate with others to obtain higher roadway capacity due to the reduction of headways. Previous studies also show that vehicle connectivity and automation can improve real-world fuel economy by facilitating eco-friendly driving style and traffic operations [137]. Some pilot applications, such as EAD and eco-speed harmonization, have been developed and deployed based on V2V and vehicle-to-infrastructure (V2I) communication, showing significant savings on fuel consumption [138]. CAV technology also incentivizes vehicle electrification and shared mobility, which further enhance the energy efficiency of transportation systems.

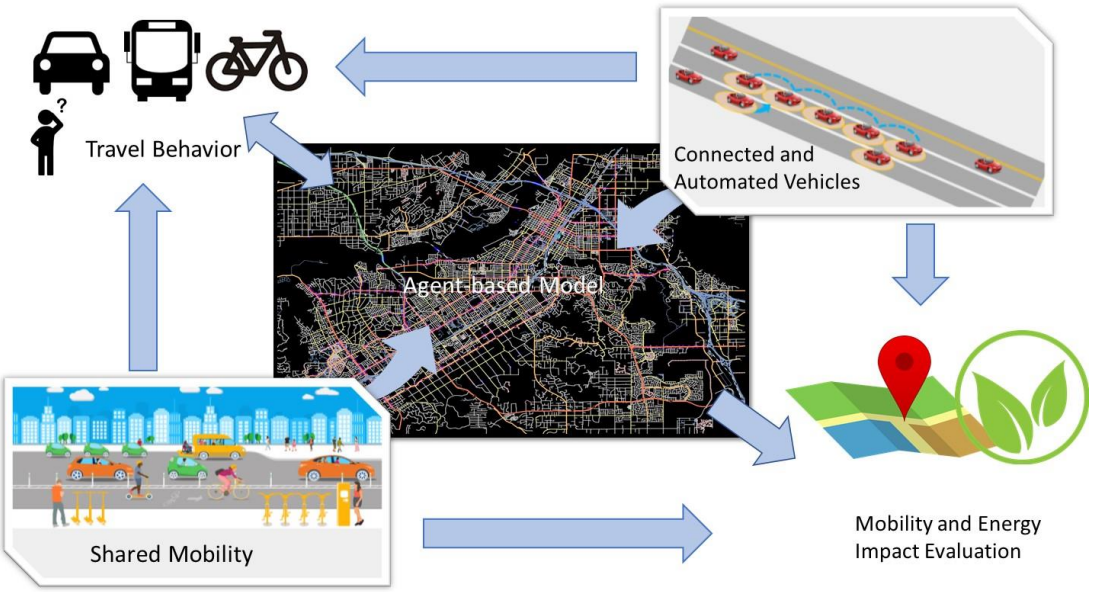
Shared mobility applications, especially those that integrate with CAV technologies (e.g. autonomous taxis), remarkably change the travel behavior and traffic demand of the current transportation system [139]. Numerical studies have shown that shared mobility would reduce the vehicle ownership, usage, and vehicle miles traveled (VMT), and therefore benefit the mobility and environmental sustainability of entire transportation system [140][141]. Carsharing and ridesharing restrain the vehicle ownership and usage

and reduce energy consumption by improving the utilization of vehicle [142]. On-demand ride services, such as ride-sourcing and e-haul services, have both positive and negative impact on the vehicle usage. They may induce additional travel but would also extend the catchment area and encourage the use of public transit.

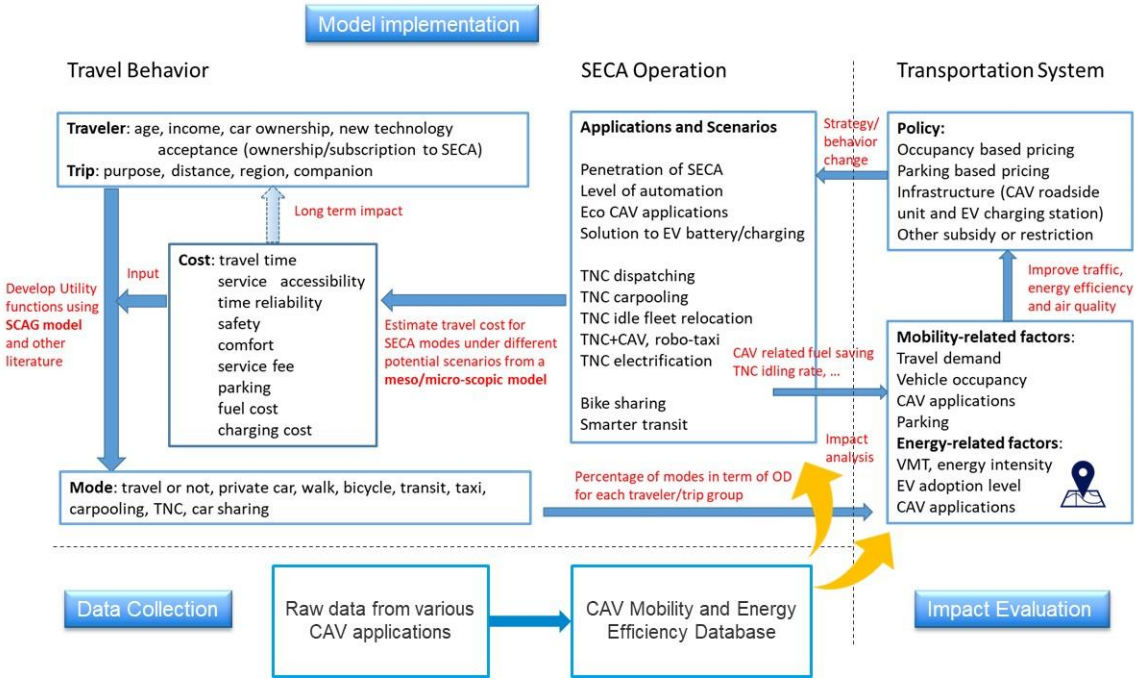
Those disruptive transportation technologies, which can summarize as Shared, Electric, Connected, and Automated (SECA) vehicles, have the potential to alter transportation costs, change the travel patterns and modes, improve or degrade the efficiency of traffic systems, and even influence the land use and residential location choices. All this will directly or indirectly affect the system-level VMT, energy consumption, air pollutant emissions and public health. To better understand and quantify the combined impact of the new mobility technologies at the transportation system level, novel models and tools have to be developed. In NCHRP Research Report 896, the framework and guidelines were proposed to help state departments of transportation (DOTs) and metropolitan planning organizations (MPOs) to upgrade their modeling and forecasting tools to adapt the impacts of CAVs on transportation supply, road capacity, and travel demand [143]. Harb et al. mimicked the potential operation mode of private owned automated vehicle by by providing 60 h of free chauffeur service for each participating household, finding a significant increase in VMT and number of trips [144]. For shared mobility, especially ride-hailing service provided by transportation network company (TNC) such as Uber and Lyft, some studies have shown its impact on traveler's behavior and transportation system efficiency. Clewlow and Mishra conducted comprehensive travel and residential survey to investigate the adoption of, use, and travel behavior impacts

of ride-hailing [145]. Circella et al. explores the factors affecting the adoption of various types of shared mobility services, with a focus on ride-hailing services [146]. Wenzel et al. studied the impact of TNC on VMT and energy use using detailed data from 1.5 million individual rides provided by RideAustin in Austin Texas. This research showed 41–90% increase on the net energy use due to the introduction of effect of ride-hailing services [147]. To the best of the authors' knowledge, there are very few researches on quantifying the combined impact of CAV and shared mobility.

In this chapter, we develop a three-phase system framework for mobility and energy efficiency evaluation considering the disruptive transportation technologies. As shown in the high-level system architecture diagram (Figure 1a), the proposed framework developed models to quantify the change of traffic operation performance and travel behavior due to the introduction of CAVs and shared mobility at different penetration and development levels. An agent-based simulation model is then developed to integrate all components in a same network of the City of Riverside. Preliminary numerical results address that impact at different CAV penetration levels and different TNC vehicle percentage. The rest of this chapter is organized as follows. In Section 6.2, we will introduce the proposed mesoscopic simulation-based framework. Section 6.3 and 6.4 will discuss the key components in the framework - CAV mobility and energy efficiency database (CAVMEED) and BEAM-in-the-loop model. Section 6.5 and 6.6 will show the model implementation and numerical result from the agent-based simulation, followed by the conclusion remarks.



(a) High-level System Architecture



(b) Three-phase Model Framework

Figure 39. System Architecture and Model Framework

6.2 Model Framework

As shown in Figure 39 (b), a three-phase system framework is developed to evaluate the mobility and energy efficiency impact of new mobility technologies to the transportation system. In this framework, the data collection phase aims to build the CAV mobility and energy efficiency database (CAVMEED) based on the real world and simulation data from various CAV applications. CAVMEED provide well-calibrated CAV-related parameters to the models and simulation networks in the other two phases. The model implementation phase integrates the discrete choice model-based travel behavior model into the traffic operation model using agent-based simulation. Below is the description of all the components in this phase.

The travel behavior level shows how travelers evaluate different travel cost factors and choose the travel mode, including: 1) Travelers and Trips: demographic information and trip purposes distribution provided by census data and regional transportation planning model, e.g. Southern California Association of Governments (SCAG) model; 2) Travel Cost: The average value of travel cost attributes (e.g. travel time, service accessibility, time reliability, safety, comfort, service fee, parking fee, fuel/charging cost) for specific origin-destination (OD) under certain scenarios; and 3) Travel Mode: the decision to make a trip or not, and the mode options the traveler may take, e.g. private car, walk, bicycle, transit, taxi, carpooling, transportation network company (TNC), or car sharing.

The Shared Electric Connected and Automated (SECA) operation level shows future traffic scenarios (in terms of market penetration, automation level, etc.) and the corresponding operation performance. We identify future transportation system scenarios

in terms of different SECA penetration and development levels and employ a mesoscopic agent-based simulation platform to accommodate all major SECA applications. The real-world data and micro-simulation data from CAVMEED are used to calibrate the parameters in the platform, e.g. link capacity and energy intensity.

The impact evaluation phase conducts analysis on mobility and energy-efficiency, and evaluation of potential policies to mitigate the negative impacts at the transportation system level. The factors that affect the performance of the transportation system, such as travel demand, vehicle occupancy and adoption of CAV applications, are integrated to analyze the impact of the new technologies on mobility, e.g. congestion level, VMT change and speed distribution. The VMT and speed bin information are then applied to the energy-intensity model, Route-E, which is calibrated from a large set of real-world drive cycles simulated in FASTSim. In this way, we evaluate the state-level energy-intensity impact of CAV technology coupled with shared mobility. Finally, we review and evaluate the policies, e.g. occupancy or parking-based pricing, to mitigate the potentially increased traffic congestion and energy consumption due to the induced travel demand and VMT.In the following sections, we will provide the details of all the components under this framework.

6.3 CAV Mobility and Energy Efficiency Database

The data collected from existing CAV applications is the foundation of the Shared Electric Connected and Automated (SECA) operation level in the model framework shown in (Figure 1b). For the CAV-impacted traffic, the major mobility-related inputs of the mesoscopic simulation platform, such as link capacities and link performance functions,

have to be recalibrated in term of the penetration of CAVs. On the other hand, some CAV applications, such as eco-approach and departure and speed harmonization, directly enhance the energy efficiency of individual vehicles. The mobility and energy impact of those applications is associated with the impact evaluation phase at the transportation system level.

There are many CAV-related factors that would impact traffic energy efficiency, including traffic demand, CAV penetration, automation level and vehicle type distribution. To collect and process sufficient and diverse data that satisfy the need of a comprehensive impact analysis, we first define five major scenarios that would cover almost all CAV applications and experiments:

1. Single CAV with no other vehicles.
2. Single CAV in mixed connected traffic.
3. Multiple CAVs in fully connected traffic.
4. Multiple CAVs in mixed connected traffic.
5. Other CAV applications with focus other than efficiency.

Note that Scenarios 1 through 4 only focus on the CAV applications that are directly associated with mobility or energy efficiency. Scenario 5 includes applications mainly developed for other purposes, such as safety and transit time reliability. For Scenario 2 and 4 (i.e. mixed traffic scenarios), the impact on equipped CAVs and the impact on other vehicles will be specified respectively. These five scenarios would well cover all major existing CAV applications, especially for those with mobility/energy benefit.

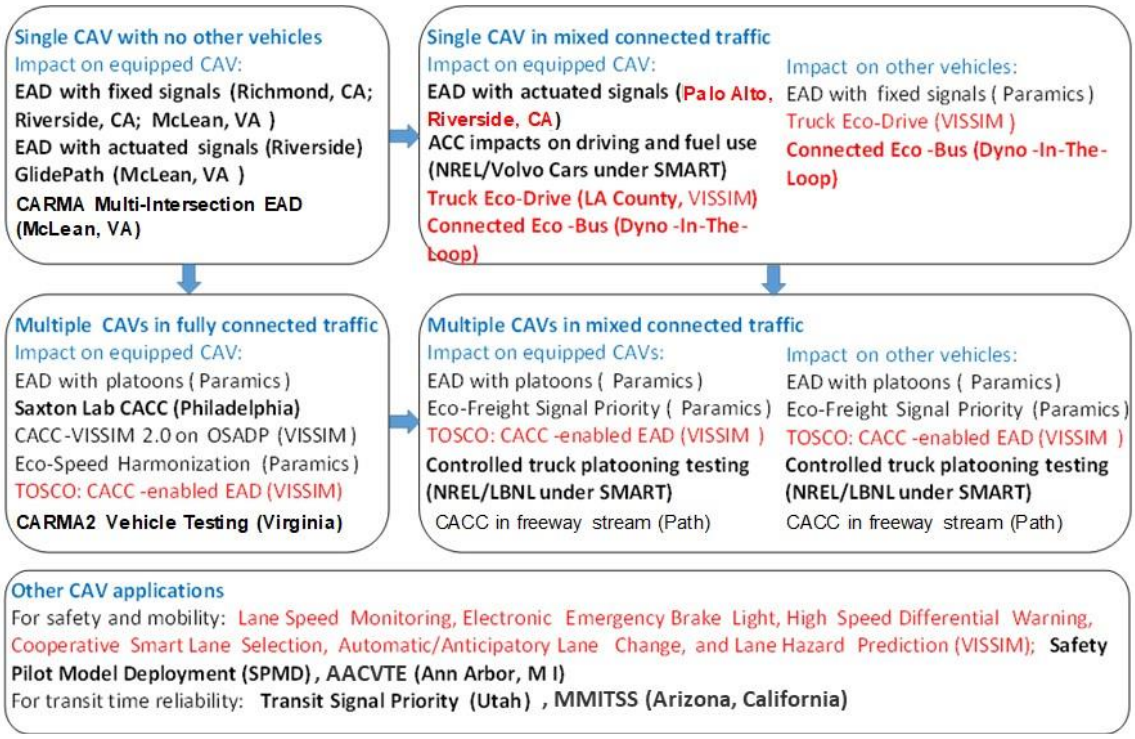


Figure 40. Major scenarios of CAV applications and experiments.

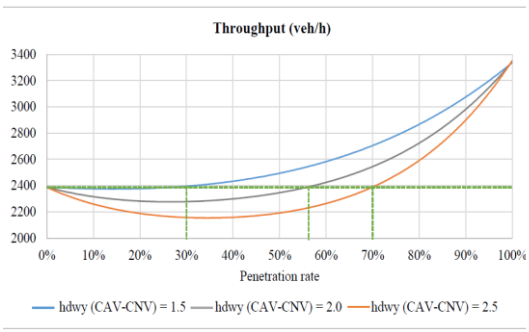
We then group the existing CAV datasets from previous field and simulation experiments into those five categories, showing their names using black font color in (Figure 2). The real-world experiments are highlighted using bold font. This figure clearly shows that the previous experiments (especially the field test) placed more emphasis on ideal cases with single vehicle and fully connected environment. The scenarios with multiple CAVs and mixed connected traffic need more experiments and data from both equipped CAVs and other conventional vehicles to support the CAV impact analysis. Therefore, the data collection phase also includes designing and implementing new experiments (highlighted in red font color) that are highly focused on multiple (and cooperative) CAVs and mixed connected traffic, which is more realistic in the transportation system of the near future. As shown in the diagram, heavy-duty vehicles,

such as buses and trucks, are another area that needs more attention. To fill in this vacancy, the research team also lead some ongoing projects, such as truck eco-drive [148] and connected eco-bus [149] to study the energy and mobility performance of heavy-duty trucks and buses due to the introduction of connectivity and automation.

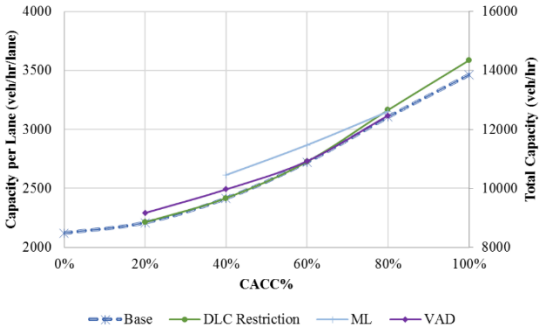
Based on the real world and simulation data from various CAV applications, we develop the CAV mobility and energy efficiency database (CAVMEED) to clean, process and archive the data and exact the key information from them to support the mesoscopic simulation model. Based on CAVMEED, two CAV-related phenomena are extensively studied: throughput improvement and trajectory smoothing effect.

Link capacity or throughput is one of the most essential input for simulation model to quantify the roadway or intersection's capability to serve more vehicles. The introduction of CAVs could enhance the link throughput by make tighter gaps between vehicles in a CACC platoon. However, in mixed connected traffic, there may be some potential loss in throughput at lower penetration rate as conventional vehicles may might keep larger headways to CAVs for safety reasons, and vice versa [150]. Based on different assumptions, different models and results are developed to describe the CAV's impact on link throughput under varying penetration rate. (Figure 3a) shows the general impact on throughput of connected autonomous vehicles considering the headway variation from different vehicle types [150]. VISSIM simulation results show that the throughput may drop if the penetration is low and the gap between CAV and conventional vehicle is assumed to be high, but with increasing penetration rates, throughput is recovered and eventually improved. Another research conducted by [151] investigated the influences of

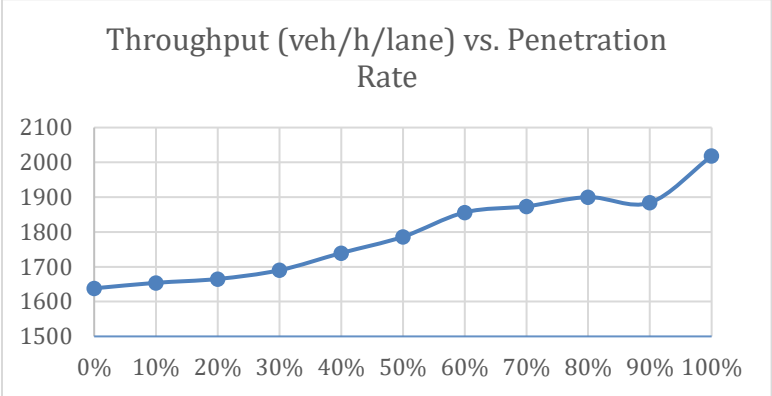
CACC operation strategies for capacity and throughput improvement in the freeway traffic stream, such as CACC managed lane (ML) strategy, vehicle awareness device (VAD) strategy and discretionary lane change (DLC) restriction. The models have been calibrated using real world data from a 13-mile corridor of the northbound SR-99 freeway near Sacramento, California. Simulation results in Figure 41 (b) shows that freeway capacity increases quadratically as the CACC market penetration increases, and the overall operational performance of the freeway corridor can be improved when the penetration is above 20%. The improvement can reach 67% at 100% penetration rate.



(a) CAV penetration vs. throughput [150]



(b) Freeway CACC penetration vs. throughput [151]



(c) Arterial CAV penetration vs. throughput

Figure 41. CAV Penetration vs. Throughput Plots.

Both model in [150] and [151] only consider the V2V communication for CAVs and ignore the impact of V2I communication on link throughput, e.g. the signal phase and

timing (SPaT) transmission to support EAD. For the CAV traffic at signalized intersection, we apply partial automation (i.e., longitudinal control) coupled with V2V/V2I communication to improve the operation of urban transportation systems in terms of mobility and environmental sustainability. This TOSCo (Traffic Optimization for Signalized Corridors) module [152] controls the logic of each vehicle in simulation, mainly based on its vehicle type, including the legacy vehicle controlled by default car-following model in VISSIM, CACC-equipped vehicles predominantly controlled by an ACC algorithm or CACC algorithm, and the TOSCo-equipped vehicles predominantly controlled by the EAD algorithm or CACC algorithm, based on the vehicle's leader/follower role in a string. VISSM network model is developed and calibrated based Riverside Innovative Corridor –University Avenue that connected Riverside Downtown and UC Riverside Campus. The control logic of TOSCo is coded in the DriverModel.dll API under VISSIM. Figure 7 shows that a combination of EAD and CACC technology would increase the capacity of intersections and have 22% throughput improvement at 100% penetration.

To evaluate the speed smoothing impact of connected EAD system, Figure 3 shows the speed distribution for test trips with informed and uninformed drivers respectively based on second-by-second instantaneous speed data [134]. With the assistance of EAD system, the percentage of low-speed mode (i.e. speed between 0~15 mph) drops significantly. Specifically, the idling or near-idling cases (i.e. speed between 0~5 mph) for the vehicle with informed driver is reduced by 22%. Those findings prove that the proposed EAD system can diminish unnecessary idling, even when the signal is actuated, and the

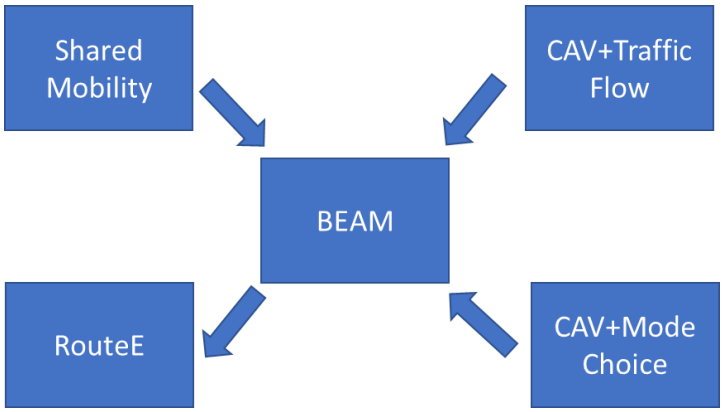
traffic condition is uncertain. Figure 5 also indicates that the EAD system reduces the percentage of relatively high-speed cases (i.e. above 30 mph, considering the speed limit of 35 mph). That means the informed driver can better control the vehicle speed to avoid unnecessary acceleration and deceleration if the SPaT message is provided.

6.4 BEAM-in-the-loop Model

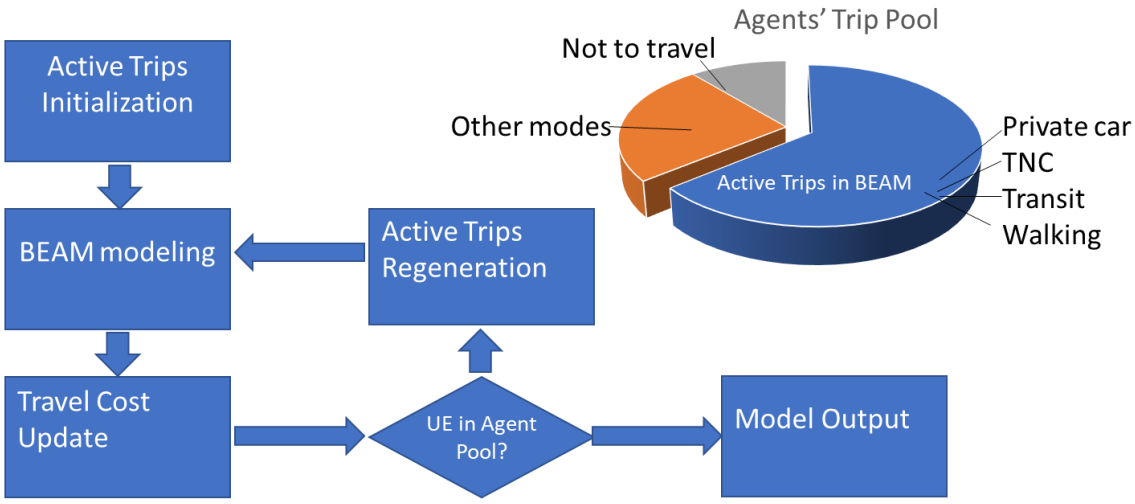
In the framework in (Figure 1b), an agent-based model that could simulate and evaluate different CAV and shared mobility scenarios is essential to the entire energy impact evaluation as it can connect traveler behavior and transportation system level and provide feasibility to quantify the impact of new mobility technologies. After comparison with other mesoscopic agent-based traffic simulator, e.g. MATSim [153] and POLARIS [154], the research team selected BEAM [155] as the main simulation platform due to its support to TNC modeling, effectiveness on large-scale network and possible synergy with its previous and ongoing work in California.

To support the model implementation, we developed three key components based on the BEAM platform: CAV-related trip generation and mode choice model to address the travel cost and travel behavior change due to CAV, CAV-related traffic flow model to quantify CAV's impact on traffic characteristics, and the shared mobility component to model ridership under TNC, car sharing or shared autonomous vehicles (SAVs). All the external models are integrated into BEAM to simulate the travel behavior and traffic condition with certain future traffic scenarios in terms of varying CAV market penetration and shared mobility operation model. The mobility efficiency can be directly outputted and summarized by BEAM from either link-based perspective or agent-based perspective. For

energy evaluation, a data-informed model named RouteE was developed to predict energy use for a certain vehicle route, which can be further utilized to estimate the energy consumption in any scale. RouteE is currently being integrated into BEAM in support of multiple SMART-Mobility tasks. (Figure 4a) shows those key components in the model implementation phase. As the CAV-related traffic flow has been introduced in the previous section, and the shared mobility module are mainly developed by the BEAM team, we will focus on the trip generation and mode choice model in this section.



(a) Major Components



(b) Flow Diagram for BEAM-in-the-loop Mode Choice Model

Figure 42. BEAM Based Modeling Approach.

BEAM provide a powerful internal engine to conduct dynamic modal choice based on the dynamic travel cost. Especially for the modes that travels over road network, such as private car, public transit and transportation network company (TNC), BEAM would update the travel cost and modal decision iteratively based on the traffic condition. However, the default modal pool of BEAM only has walking as the representative of active transportation. Moreover, BEAM does not have any mode to represent the intended travelers who opt not to travel as the travel cost (either time or money) is beyond their acceptance range, which would be the key factor to explain the induced demand by new mobility technology. Therefore, we developed a hybrid model which combines the default modal choice module of BEAM and an external discrete choice model for the modes that is not represented in BEAM. (Figure 4b) shows the flow diagram for modal choice modeling in this project. The trip pool of all the agents is first created to store all the intended trips from all travelers. An external discreate choice model is then applied to separate active trips in BEAM (e.g. private vehicle, transit, TNC and walking) from other modes and not-to-travel choice. We then apply the active trips to BEAM for agent-based simulation, and update travel cost from BEAM's output. The updated travel cost information is then loaded into the discrete choice again to check if user's equilibrium is met. If so, the modal choice results and BEAM output are adopted for evaluating the impact of CAV and shared mobility on modal choice. Otherwise, we start another iteration.

6.5 Implementation of City of Riverside Network

We developed BEAM model for the City of Riverside with over 24,000 agents using Southern California Association of Government (SCAG) data as the input of travel

demand and estimate travel activities. To build the simulation environment, BEAM require the following datasets as the input:

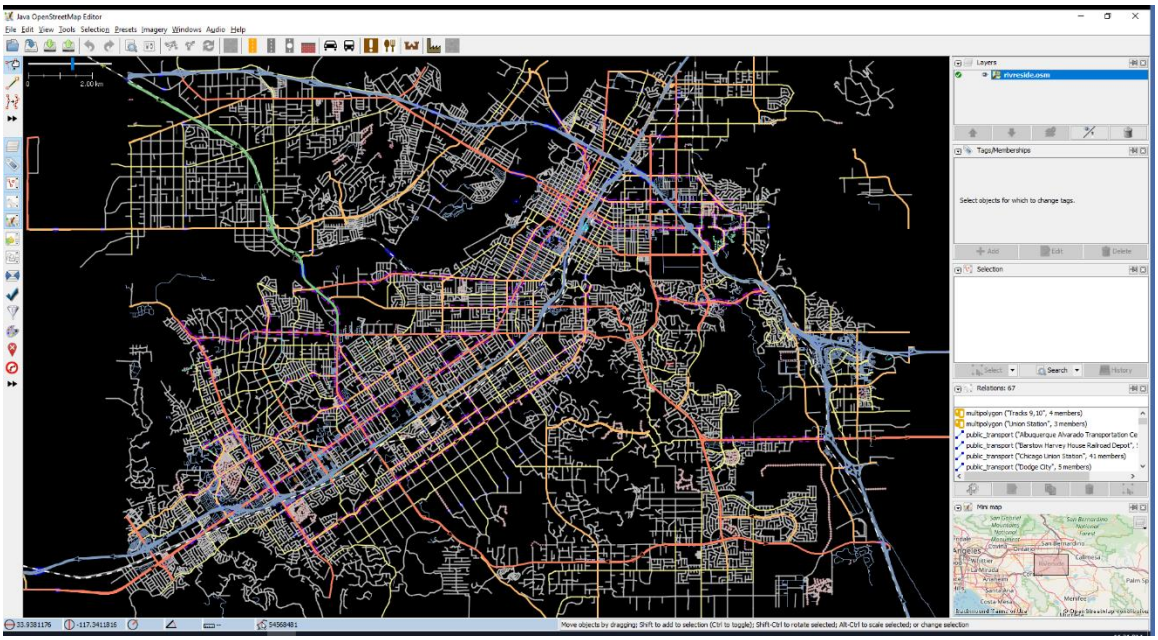
Table 19. The Input Files Required for Building Customized BEAM Model.

File name	Caption
riverside.conf	BEAM configuration file.
population.xml	Agent travel activity file.
populationAttributes.xml	Agent parameters (e.g. value of time) [optional].
households.xml	The agent and vehicle per household.
householdAttributes.xml	The household type and location information.
vehicles.xml	Vehicle types.
riverside.osm	The open street map network.
bus.zip	GTFS archives, one for each transit agency.

The riverside network is derived from OpenStreetMap.com, as shown in Figure 43 (a). The population activity is the most important and difficult input for customizing the BEAM model. It defines the travel activities of all the population in the simulated area, which should be generated by the local agent based microscopic travel demand model. The trip-based transportation model by Southern California Association of Government (SCAG) is used to synthesize the travel activities [156]. This model can output aggregated travel demand in a mesoscopic resolution (TAZ level OD pairs). The parameters are shown in Table 20.

Table 20. The Parameter List in the SCAG Trip-based Travel Activity Model.

Parameter	Definition	Sample
RSD	Orig_TAZID	43121000
RSD1	Dest_TAZID	43121000
DA	Drive Alone (# trips)	66.95
SR2_HOV	Share ride 2 persons HOV	3.85
SR3_HOV	Share ride 3 persons HOV	2.57
LHDT	Light Heavy Duty Trucks	0.44
MHDT	Medium Heavy Duty Trucks	0.22
HHDT	Heavy Heavy Duty Trucks	0.1
SR2_NONHOV	Share ride 2 persons non-HOV	11.01
SR3_NONHOV	Share ride 3 persons non-HOV	7.35
Total	total number	92.49



(a) Riverside Network from OpenStreetMap.com

(b) Riverside Network from Via – BEAM visualizer

Figure 43. City of Riverside Network.

To obtain the agent-based activates from the OD table, an activity generation algorithm is developed to produce schedule. The census block population distribution data is utilized in this algorithm. Associated with the activity generation algorithm, a Python supported XML writing package called "lxml" is also used for encoding the activity to .xml format. Algorithm 1 shown below is the pseudo code for the activity synthesize model.

The travel activity demand is generated by the TAZ based OD pairs. First, the total number of agents that travels between each (TAZ1, TAZ2) origin-destination pair is calculated by summing up the trips weighted by the number of travelers for each trip; Then, two trips (one trip to work in the morning peak hour 6:00 am to 9:00 am, and one trip back home in the afternoon peak hour 3:00 pm to 7:00 pm) are generated for each agent. The trip origin and destination are assigned randomly to census blocks in the origin and

destination TAZ, weighted by each block's population. The trip start times are assigned randomly in the morning and after peak hour range.

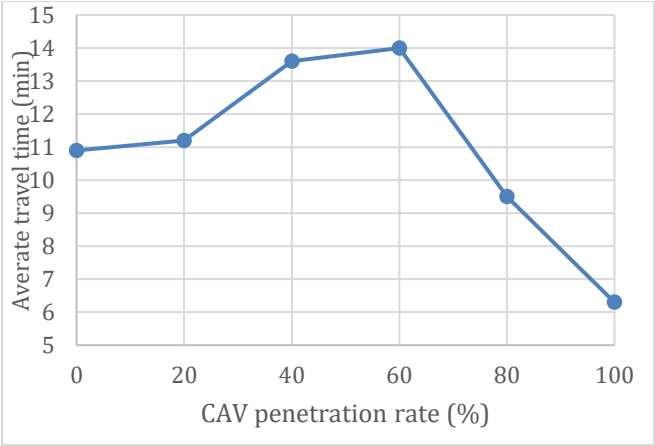
Regardless the simplicity of the activity synthesize model, it offers the required input for the BEAM model and allows a test run of the customized Riverside scenario. A more realistic and precise travel demand model will be applied using PopGen and ActivitySim. The household information was generated by assigning each agent a home and working location based on their trip to work and trip back home origin-destination pair. If all the input files are well-prepared, we load them to BEAM along with the simulation network and generate output file after the run. A software named Via is then used to visualize the network and vehicle position as shown in Figure 43 (b).

6.6 Results and Discussion

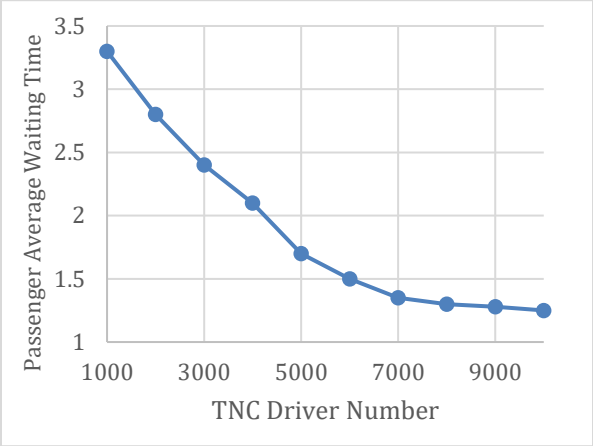
Figure 41 in shows the impact of CAVs penetration rate on the traffic throughput at different types of facility. It offers the possibility to simulate the system-level CAV impact in the BEAM by adjusting the road capacity of the road network, which can be realized by tuning the parameter in the BEAM-integrated Java Discrete Event Queue Simulator (JDEQSim) from the MATSim framework.

In Figure 44 (a), the CAV impact on average travel time in Riverside network is depicted based on the assumption in [150]. This figure indicates that the vehicle average travel time will initially increase when the CAV penetration starts increasing from zero if larger headways between CAVs and conventional vehicles are kept for safety reasons. The vehicle average travel time will then decrease in a faster manner as the penetration rate

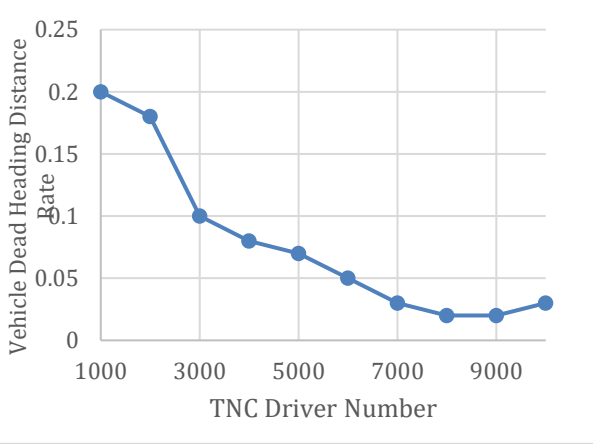
increase over 60%. This indicates that high penetration rate of CAVs will eventually improve the traffic condition by regulating the vehicle headway and increase the road capacity. Note that this result corresponds to the U-shape CAV penetration-throughput curve in [150], if we adapt the assumption in [151] or the proposed model in Figure 44 (c), the average travel time will also show different trend. It is also worth noting that the travel demand of the simulation scenario stays static during the CAV penetration rate impact analysis. The system is not considering the travel demand change brought by CAV implementations. Since more convenient and travel-time saving travel mode may increase the demand. This part will be future integrated in the analysis.



(a) The CAVs' impact on average travel time.



(b) TNC penetration vs. average waiting time



(c) TNC penetration vs. vacant travel distance

Figure 44. Preliminary Simulation Results in BEAM.

Ride hailing vehicles are emerging shared mobility forms and are already changing the mobility landscape and as driverless vehicles come online, the economics of these services will improve substantially. In BEAM, ride hailing vehicles are modeled as a fleet of taxis controlled by a centralized manager that responds to requests from customers and dispatches vehicles accordingly. To study the impact of the ride hailing vehicles to the transportation system, a series of sensitivity analysis is designed and implemented in the Riverside scenario. Figure 44 (b) shows how the penetration rate of ride hailing drivers impacts the passengers average waiting time. The overall trend indicates that, as the ride hailing penetration rate increased among the agents, more vehicles could be potentially served for ride hailing services and thus less waiting time will be needed for passengers. Figure 44 (c) shows how the penetration rate of ride hailing drivers impacts the portion of vacant travel distance for ride hailing vehicles. As the ride hailing penetration rate increased among the agents, the vehicles' vacant travel distance rate decreases.

BEAM is a mesoscopic simulator which performs well in representing network level demand-supply dynamics, in a city like Riverside or San Francisco. However, it is difficult, if not impossible, to represent all the details of a mesoscopic model at the state level. Therefore, the research team has developed an approach to extract information from highly detailed regional level model and extrapolate to the state level. This approach is inspired by [157], where they have defined several household clusters using their socio-demographic information. They assumed that the travel behavior of individuals within each cluster is homogenous and, therefore, were able to transfer cluster membership and

behaviors to national level data such as the National Household Travel Survey (NHTS) [158].

To perform the state-level extrapolation, the agent level simulated energy efficiency can be classified to homogenous agent groups by their (1) socio-demographic characteristics (2) household variables, and (4) network level attributes for the home base. Socio-demographic attributes such as age and sex are already a part of the mesoscopic model. Several household related attributes such as household income, number of vehicles available etc. are included in the household descriptor of the model. The research team is currently working on extracting the network level attributes such as presence of public transit, highway miles per unit area, population density from the network defined in both the San Francisco and the Riverside model.

The research team has analyzed the 2017 National Household Transportation Survey (NHTS) California database, which is currently maintained by the Transportation Secure Data Center (TSDC) at the National Renewable Energy Laboratory. The survey includes 26,095 California households and recorded a total of 55,793 participants. In addition, the research team is currently working on a statewide network data collected from the California Department of Transportation (CalTrans) GIS data library.

The scaling algorithm will be implemented by estimating the socio-demographic, household, and network attributes for each of the 55,793 respondents of the California NHTS survey. The state level network attributes will be estimated at the level of census tract. Since the NHTS dataset as maintained by NREL has spatial reference for the home

location, it will be possible to link a survey respondent to the corresponding network attribute.

Conclusions

In this chapter, a mesoscopic simulation-based framework to quantifying the combined impact of CAV and shared mobility is proposed. Under the proposed framework, multiple models are developed to quantify the change of traffic operation performance and travel behavior due to the introduction of SECA applications at different penetration and development levels. This research includes three phases – data collection, model implementation and impact evaluation. We collected data from vehicles with CAV and shared mobility technologies (primarily deployed in California) and build CAVMEED for other CAV studies. We built Riverside model, the first BEAM model in Southern California, to implement applications that are associated with CAV and shared mobility. Preliminary numerical results show the relationship between CAV penetration levels and average travel time, and the connection between TNC penetration and waiting time and vacancy rate. This research addresses key barriers by quantifying the transportation system-wide mobility, energy and behavior impacts from new mobility technologies using real-world data. Future direction of research includes:

1. Further fine-tune the parameters in the proposed mode choice model, and develop discrete choice models to characterize the induced demand effect.

2. Define multiple scenarios that could represent the various stages of new mobility technologies in the future.

3. Evaluate the regional mobility, VMT, and energy impact of CAV and shared mobility for Riverside and Bay Area using the proposed BEAM-based model.

4. Evaluate the potential effects of proposed policies to mitigate adverse energy outcomes.

Chapter 7
Conclusions and Future Work

7.1 Conclusions of the Dissertation

The emerging challenges and opportunities brought by shared mobility and CAVs attracts significant research interests and has great potential to benefit the current transportation system in terms of safer, faster and more environment friendly travels. Meanwhile, increasing volume of data has been generated for logging, monitoring, and perception purposes, which contains valuable information and knowledge that could help accelerate this process. Therefore, it is essential to understand the ITS data from ride sharing and autonomous driving to help with the development of high-level applications.

In this dissertation research, a framework for knowledge discovery and data mining strategies to facilitate the shared mobility and CAV applications is proposed. The main sources of data obtained from shared mobility and autonomous driving are analyzed and mined for transportation knowledge for application development. The practicality of multiple data-driven methods for shared automated mobility applications are proven with case study and result discussions. The state-of-art machine learning methods are also customized and adopted to the cases, which gives example and indicates the great potential

of the crossover study of AI and ITS. Specifically, our dissertation is formed in a high-to-low-level structure, with the following achievement:

- A data mining method for road infrastructure mapping that extracts intersection and stop bar locations from GPS trajectories. The method includes a novel entropy-based analysis for intersection identification and a modified constrained Gaussian mixture model (CGMM) for stop bar position estimation.
- A data-driven framework for ride-hailing services demand prediction and fleet operation optimization with help of historical data mining. Long short-term memory (LSTM) network and convolutional neural network (CNN) are customized to predict the short-term passenger demand. The supply-demand optimal matching problem is formulated into several math models and solved by reinforcement learning. An end-to-end advanced driving assistant system by learning driving behavior from video data. This helps with the decision making for autonomous driving vehicles so that they behave more like human. A case study of lane change behavior prediction and lane localization is conducted and proves the practical accuracy and responding time.
- A comprehensive multi-agent transportation simulation for shared mobility and CAV applications with help of the resident's activity data, which offers the insight of the future impacts of the applications would bring on the Southern California transportation system.

7.2 Selected Publications Resulting from this Research

[1] "Intersection and stop bar position extraction from crowdsourced GPS trajectories", *Transportation Research Board 96th Annual Meeting,* Washington D.C., Jan. 2017.

[2] "Intersection and Stop Bar Position Extraction from Vehicle Positioning Data", *IEEE Transactions on Intelligent Transportation Systems.*

[3] "Predicting the Number of Uber Pickups by Deep Learning," *Transportation Research Board 97th Annual Meeting,* Washington D.C., Jan. 2018.

[4] "Data-Driven Multi-step Demand Prediction for Ride-hailing Services Using Convolutional Neural Network," *Computer Vision Conference (CVC) 2019 - SAI Conferences.*

[5] "An Innovative Framework to Evaluate the Performance of Connected Vehicle Applications: From the Perspective of Speed Variation-Based Entropy (SVE), " IEEE Intelligent Transportation Systems Magazine.

[6] "Agent-Based Modeling and Simulation of Connected and Automated Vehicles Using Game Engine: A Cooperative On-Ramp Merging Study," *Transportation Research Board 98th Annual Meeting,* Washington D.C., Jan. 2019.

[7] "Eco-Approach and Departure along Signalized Corridors," *Transportation Research Board 98th Annual Meeting,* Washington D.C., Jan. 2019.

[8] "Vision-Based Lane-Changing Behavior Detection Using Deep Residual Neural Network" *IEEE 22th International Conference on Intelligent Transportation Systems,* Nov. 2019.

[9] "Evaluating the environmental impact of traffic congestion based on sparse mobile crowd-sourced data," *2017 IEEE Conference on Technologies for Sustainability (SusTech), 1-6.*

[10] "Driver Behavior Modeling using Game Engine and Real Vehicle: A Learning-Based Approach," *IEEE Transactions on Intelligent Vehicles.*

7.3 Future Work

Although many positive results have been achieved in this dissertation, there are still several open problems that need to be addressed in future work related to the data-driven methods and techniques supported shared mobility and CAV applications.

First, a comprehensive data fusion framework and platform is necessary to designed and developed to best utilize all kinds of data collected in transportation system and in-vehicle or mobile devices. Future shared mobility and CAV applications will rely on sensing data from multiple sources for more accurate decision making to improve safety, mobility and energy consumption. For example, traffic and road infrastructure information can be good context information to facilitate the perception data stream for advanced driving assistant system and self-driving vehicles. Challenges for the data fusion framework includes but not limited to processing speed, complex pattern recognition, and ensemble learning. Therefore, in the future research effort may be conducted for a data fusion framework for real-time high efficiency data fusion algorithms.

Second, identifying and closing the gap between theoretical research and experimental implementation for the proposed algorithms and applications in the thesis is another future work direction. It is true that many advanced methodologies have been proposed and analyzed in theory, however, the gap between theoretically functional and practically functional needs to be identified and closed. For example, the theoretical studies of CAV systems in most cases ignored the destabilizing effect of communication latency. Shared mobility and CAV applications that would appear to be stable based on the theoretical analyses are not always stable in practical implementations due to unavoidable latencies in communications. In addition, theoretical research results need to be tested under various realistic conditions to identify this gap, but that could be both labor-intensive and time-consuming.

Finally, it is important to develop more ready-to-market applications with field test validation. The shared mobility and CAV applications developed in this dissertation made

are validated with numerical analysis, simulation, and historical data validation. However, it is essential that real-world tests are conducted before the promising benefit can be fully proved. Based on the developed knowledge discovery and data mining framework, comprehensive design of the application should be able to ensure performance with high volume data in real-time. For example, the real-time ride-hailing demand prediction and fleet dispatching system needs to consider more constraints in real-world implementation, including traffic condition, human execution error, vehicle refueling or charging, etc. In order to facilitate more ready-to-market shared mobility and CAV applications, the future research and development might take advantage of more complex simulation and analysis as well as real-world in-loop tests.

BIBLIOGRAPHY

[1] E. Mazareanu, "Ride-sharing services in the U.S.," *www.statista.com*. [Online]. Available: https://www.statista.com/topics/4610/ridesharing-services-in-the-us [Accessed: 14-Feb-2020].

[2] "Ride Hailing - United States Statista," *www.statista.com.* [Online]. Available: https://www.statista.com/outlook/368/109/ride-hailing/united-states [Accessed: 14-Feb-2020].

[3] I. Wagner, "Autonomous vehicle technology - Statistics & Facts," *www.statista.com*. [Online]. Available: https://www.statista.com/topics/3573/autonomous-vehicle-technology [Accessed: 14-Feb-2020].

[4] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected Vehicles: Solutions and Challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, 2014.

[5] J. Fiosina and M. F. J. P. Müller, "Big Data Processing and Mining for Next Generation Intelligent Transportation Systems," Jurnal Teknologi, vol. 63, no. 3, Feb. 2013.

[6] Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P. and Uthursamy, R. Advances in knowledge discovery and data mining, AAAI Press/The MIT Press, Cambridge, MA, 1996.

[7] Kohavi, R. Data mining and visualization. In: Sixth Annual Symposium on Frontiers of Engineering, National Academy Press, D. C., 2001, p. 30-40.

[8] J. Farrell, M. Todd, and M. Barth. "Best practices for surveying and mapping roadways and intersections for connected vehicle applications." (2016).

[9] M. Thuy and F. León, "Lane Detection and Tracking Based on Lidar Data," *Metrology and Measurement Systems*, vol. 17, no. 3, Jan. 2010.

[10] B. Yang, Z. Dong, G. Zhao, and W. Dai, "Hierarchical extraction of urban objects from mobile laser scanning data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 99, pp. 45–57, 2015.

[11] "Interview with Robert Mankowski, Vice President, Digital Cities, Bentley Systems," *The magazine for civil & structural engineers*. [Online]. Available: https://informedinfrastructure.com/14059/mobile-lidar-proves-cost-effective-for-pavement-mapping. [Accessed: 07-Nov-2018].

[12] X. Guan and H. Wu, "Leveraging the power of multi-core platforms for large-scale geospatial data processing: Exemplified by generating DEM from massive LiDAR point clouds," *Computers & Geosciences*, vol. 36, no. 10, pp. 1276–1282, 2010.

[13] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[14] P. Hao, K. Boriboonsomsin, C. Wang, G. Wu, and M. Barth, "Connected eco-approach and departure (EAD) system for diesel trucks," in *Transportation Research Board 97th Annual Meeting*, 2018, No. 18-06464.

[15] M. Amo, F. Martinez, and M. Torre, "Road extraction from aerial images using a region competition algorithm," *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1192–1201, 2006.

[16] J. Hu, A. Razdan, J. C. Femiani, M. Cui, and P. Wonka, "Road Network Extraction and Intersection Detection From Aerial Images by Tracking Road Footprints," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 4144–4157, 2007.

[17] Y.-W. Seo, C. Urmson, and D. Wettergreen, "Exploiting publicly available cartographic resources for aerial image analysis," *Proceedings of the 20th International Conference on Advances in Geographic Information Systems - SIGSPATIAL 12*, 2012.

[18] J. Jung and S.-H. Bae, "Real-Time Road Lane Detection in Urban Areas Using LiDAR Data," *Electronics*, vol. 7, no. 11, p. 276, 2018.

[19] P. Jiangui and G. Guang, "A method for main road extraction from airborne LiDAR data in urban area," *2011 International Conference on Electronics, Communications and Control (ICECC)*, 2011.

[20] H. Guan, J. Li, S. Cao, and Y. Yu, "Use of mobile LiDAR in road information inventory: a review," *International Journal of Image and Data Fusion*, vol. 7, no. 3, pp. 219–242, 2016.

[21] M. Yadav and A. K. Singh, "Rural Road Surface Extraction Using Mobile LiDAR Point Cloud Data," *Journal of the Indian Society of Remote Sensing*, vol. 46, no. 4, pp. 531–538, 2017.

[22] M. Lehtomäki, A. Jaakkola, J. Hyyppä, A. Kukko, and H. Kaartinen, "Detection of Vertical Pole-Like Objects in a Road Environment Using Vehicle-Based Laser Scanning Data," *Remote Sensing*, vol. 2, no. 3, pp. 641–664, 2010.

[23] Z. Liu, J. Wang, and D. Liu, "A New Curb Detection Method for Unmanned Ground Vehicles Using 2D Sequential Laser Data," *Sensors*, vol. 13, no. 1, pp. 1102–1120, 2013.

[24] P. Hao, C. Wang, G. Wu, K. Boriboonsomsin. and M. Barth, "Evaluating the environmental impact of traffic congestion based on sparse mobile crowd-sourced data," in *4th International IEEE Conference on Technologies for Sustainability (SusTech)*, 2017, pp. 1-6.

[25] C. Wang, P. Hao, G. Wu, X. Qi, and M. Barth, "Predicting the Number of Uber Pickups by Deep Learning," in *Transportation Research Board 97th Annual Meeting*, 2018, No. 18-06738.

[26] C. Wang, Y. Hou, and M. Barth, "Data-Driven Multi-step Demand Prediction for Ride-Hailing Services Using Convolutional Neural Network," in *Science and Information Conference*, pp. 11-22. Springer, Cham, 2019.

[27] S. Worrall and E. Nebot, "Automated process for generating digitised maps through GPS data compression," *Australasian Conference on Robotics and Automation*, 2007.

[28] S. Edelkamp and S. Schrödl, "Route Planning and Map Inference with Global Positioning Traces," *Lecture Notes in Computer Science Computer Science in Perspective*, pp. 128–151, 2003.

[29] L. Cao and J. Krumm, "From GPS traces to a routable road map," Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS 09, 2009.

[30] A. Schindler, "Vehicle self-localization with high-precision digital maps," *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[31] "Explore the Uber Platform | Uber United States." [Online]. Available: https://www.uber.com/. [Accessed: 19-Mar-2020].

[32] Lyft, Inc, "Become a Driver or Get a Ride Now," *Lyft*. [Online]. Available: https://www.lyft.com/. [Accessed: 19-Mar-2020].

[33] "DiDi Chuxing Platform Official Website,". [Online]. Available: https://www.didiglobal.com/. [Accessed: 19-Mar-2020].

[34] "Ride Austin," *Ride Austin*. [Online]. Available: http://www.rideaustin.com/. [Accessed: 19-Mar-2020].

[35] H. W. Chang, Y. C. Tai, and J. Y. J. Hsu, "Context-aware taxi demand hotspots prediction," *International Journal of Business Intelligence and Data Mining*, vol. 5, no. 1, pp. 3-18, 2010.

[36] L. Moreira-Matias, J. Gama, M. Ferreira, and L. Damas, "A predictive model for the passenger demand on a taxi network," In Proc. *15th International IEEE Conference on Intelligent Transportation Systems*, Anchorage, Alaska, USA, 2012.

[37] Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., & Damas, L. (2013). Predicting taxi–passenger demand using streaming data. IEEE Transactions on Intelligent Transportation Systems, 14(3), 1393-1402.

[38] Y. Gong, B. Fang, S. Zhang, and J. Zhang, "Predict New York city taxi demand," NYC Data Science Academy, 2016. Available: https://blog.nycdatascience.com/student-works/predict-new-york-city-taxi-demand/

[39] J. Ke, H. Zheng, H. Yang, and X. Chen, "Short-term forecasting of passenger demand under on-demand ride services: a spatio-temporal deep learning approach," *Transportation Research Part C*, vol. 85, pp. 591-608, 2017.

[40] Wang, C., Hao, P., Wu, G., Qi, X. and Barth, M., 2018. *Predicting the Number of Uber Pickups by Deep Learning* (No. 18-06738).

[41] Xu, J., Rahmatizadeh, R., Bölöni, L., & Turgut, D. (2017). Real-Time Prediction of Taxi Demand Using Recurrent Neural Networks. IEEE Transactions on Intelligent Transportation Systems.

[42] Liao, S., Zhou, L., Di, X., Yuan, B., & Xiong, J. (2018, January). Large-scale short-term urban taxi demand forecasting using deep learning. In Proceedings of the 23rd Asia and South Pacific Design Automation Conference (pp. 428-433). IEEE Press.

[43] Matsugu, M., Mori, K., Mitari, Y. and Kaneda, Y., 2003. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, *16*(5-6), pp.555-559.

[44] LeCun, Yann. "LeNet-5, convolutional neural networks". Retrieved 16 November 2013.

[45] Borkar, M. Hayes, M. T. Smith, A Novel Lane Detection System With Efficient Ground Truth Generation. *IEEE Trans. Intelligent Transportation Systems*, vol. 13, no. 1, pp. 365-374, 2012.

[46] H. Deusch, J. Wiest, S. Reuter, M. Szczot, M. Konrad, K. Dietmayer, A random finite set approach to multiple lane detection. *ITSC*, pp. 270-275, 2012.

[47] K.-Y. Chiu, S.-F. Lin, Lane detection using color-based segmentation. *Intelligent Vehicles Symposium*, pp. 706-711, 2005.

[48] H. Loose, U. Franke, C. Stiller, Kalman particle filter for lane recognition on rural roads. *Intelligent Vehicles Symposium*, pp. 60-65, 2009.

[49] Z. Teng, J.-H. Kim, D.-J. Kang, Real-time Lane detection by using multiple cues. *Control Automation and Systems*, pp. 2334-2337, 2010.

[50] A. L´opez, J. Serrat, C. Canero, F. Lumbreras, T. Graf, Robust lane markings detection and road geometry computation. *International Journal of Automotive Technology*, vol. 11, no. 3, pp. 395-407, 2010.

[51] G. Liu, F. W¨org¨otter, I. Markelic, Combining Statistical Hough Transform and Particle Filter for robust lane detection and tracking. *Intelligent Vehicles Symposium*, pp. 993-997, 2010.

[52] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, H. Chen, A novel lane detection based on geometrical model and Gabor filter. *Intelligent Vehicles Symposium*, pp. 59-64, 2010.

[53] Z. Kim, Robust Lane Detection and Tracking in Challenging Scenarios. *IEEE Trans. Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16-26, 2008.

[54] R. Danescu, S. Nedevschi, Probabilistic Lane Tracking in Difficult Road Scenarios Using Stereovision. *IEEE Trans. Intelligent Transportation Systems*, vol. 10, no. 2, pp. 272-282, 2009.

[55] Z. Teng, J.-H. Kim, D.-J. Kang, Real-time Lane detection by using multiple cues. Control Automation and Systems, pp. 2334-2337, 2010.

[56] A. Bar-Hillel, R. Lerner, D. Levi, G. Raz, Recent progress in road and lane detection: a survey. Mach. Vis. Appl., vol. 25, no. 3, pp. 727-745, 2014.

[57] R. Gopalan, T. Hong, M. Shneier, R. Chellappa, A Learning Approach Towards Detection and Tracking of Lane Markings. *IEEE Trans. Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1088-1098, 2012.

[58] J. Kim, M. Lee, Robust Lane Detection Based On Convolutional Neural Network and Random Sample Consensus. ICONIP, pp. 454-461, 2014.

[59] 16. B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, A. Y. Ng, An Empirical Evaluation of Deep Learning on Highway Driving. CoRR abs/1504.01716, 2015.

[60] B. He, R. Ai, Y. Yan, X. Lang, Accurate and robust lane detection based on Dual-View Convolutional Neutral Network. *Intelligent Vehicles Symposium*, pp. 1041-1046, 2016.

[61] J. Li, X. Mei, D. V. Prokhorov, D. Tao, Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene. *IEEE Trans. Neural Netw. Learning Syst*., vol. 28, no.3, pp. 690-703, 2017.

[62] S. Lee, J.-S. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. S. Hong, S.-H. Han, I. S. Kweon, VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition. CoRR abs/1710.06288, 2017.

[63] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[64] Pascanu, R., Mikolov, T., & Bengio, Y. (2013, February). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310-1318).

[65] Viola, P., Jones, M. J., & Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, *63*(2), 153-161.

[66] Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, *29*(1), 51-59.

[67] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). IEEE.

[68] Wiesel, T. N., & Hubel, D. H. (1963). Single-cell responses in striate cortex of kittens deprived of vision in one eye. *Journal of neurophysiology*, *26*(6), 1003-1017.

[69] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[70] Goodale, M. A., & Milner, A. D. (1992). Separate visual pathways for perception and action.

[71] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, and C. Wilson, “Mining GPS Traces for Map Refinement,” *Data Mining and Knowledge Discovery*, vol. 9, no. 1, pp. 59–87, 2004.

[72] L. Tang, X. Yang, Z. Dong, and Q. Li, “CLRIC: Collecting Lane-Based Road Information Via Crowdsourcing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2552–2562, 2016.

[73] E. Uduwaragoda, A. Perera, and S. Dias, “Generating lane level road data from vehicle trajectories using Kernel Density Estimation,” *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013.

[74] Y. Chen and J. Krumm, “Probabilistic modeling of traffic lanes from GPS traces,” Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS 10, 2010.

[75] K. Krippendorff, “Mathematical Theory of Communication,” *Encyclopedia of Communication Theory*.

[76] G. M. Weiss, “Data Mining in Telecommunications,” *Data Mining and Knowledge Discovery Handbook*, pp. 1189–1201.

[77] “Update: GNSS Accuracy: Lies, Damn Lies, and Statistics,” *GPS World*, 10-Aug-2018. [Online]. Available: https://www.gpsworld.com/gpsgnss-accuracy-lies-damn-lies-and-statistics-1134/. [Accessed: 07-Nov-2018].

[78] “Normal distribution,” *Wikipedia*, 23-Oct-2019. [Online]. Available: https://en.wikipedia.org/wiki/Normal_distribution. [Accessed: 07-Nov-2018].

[79] “Sum of normally distributed random variables,” *Wikipedia*, 29-May-2019. [Online]. Available: https://en.wikipedia.org/wiki/Sum_of_normally_distributed_random_variables. [Accessed: 07-Nov-2018].

[80] R. J. Little and D. B. Rubin, “Wiley Series in Probability and Statistics,” *Statistical Analysis with Missing Data Wiley Series in Probability and Statistics*, pp. 383–389, 2014.

[81] X.-L. Meng and D. B. Rubin, “Maximum likelihood estimation via the ECM algorithm: A general framework,” *Biometrika*, vol. 80, no. 2, pp. 267–278, 1993.

[82] R. M. Neal and G. E. Hinton, “A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants,” *Learning in Graphical Models*, pp. 355–368, 1998.

[83] *Intelligent Transportation Systems - Safety Pilot Model Deployment Data*. [Online]. Available: https://www.its.dot.gov/factsheets/safetypilot_modeldeployment.htm. [Accessed: 07-Nov-2017].

[84] N. Williams, G. Wu, and P. Closas, “Impact of positioning uncertainty on eco-approach and departure of connected and automated vehicles,” *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018.

[85] Min, W. and L. Wynter, Real-time road traffic prediction with spatio-temporal correlations. Transportation Research Part C: Emerging Technologies, Vol. 19, No. 4, 2011, pp. 606–616.

[86] Lv, Y., Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, Vol. 16, No. 2, 2015, pp. 865–873.

[87] Chen, C., D. Zhang, Z.-H. Zhou, N. Li, T. Atmaca, and S. Li, B-Planner: Night bus route planning using large-scale taxi GPS traces. In Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on, IEEE, 2013, pp. 225–233.

[88] Zhan, X., X. Qian, and S. V. Ukkusuri, A graph-based approach to measuring the efficiency of an urban taxi service system. IEEE Transactions on Intelligent Transportation Systems, Vol. 17, No. 9, 2016, pp. 2479–2489.

[89] Li, B., D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang, Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 63–68.

[90] Zheng, Y., F. Liu, and H.-P. Hsieh, U-Air: When urban air quality inference meets big data. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2013, pp. 1436–1444.

[91] Moreira-Matias, L., J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, Predicting taxi-passenger demand using streaming data. IEEE Transactions on Intelligent Transportation Systems, Vol. 14, No. 3, 2013, pp. 1393–1402.

[92] Fivethirtyeight, Uber Is Serving New York's Outer Boroughs More Than Taxis Are. https://fivethirtyeight.com/features/uber-is-serving-new-yorks-outer-boroughs-more-than-taxis-are/, 2015, accessed: 2017-03-28.

[93] Fivethirtyeight, Public Transit Should Be Uber's New Best Friend. https://fivethirtyeight.com/features/public-transit-should-be-ubers-new-best-friend/, 2015, accessed: 2017-03-28.

[94] Wikipedia, Decision tree learning. https://en.wikipedia.org/wiki/Decision_tree_learning, 2017, accessed: 2017-04-30.

[95] Friedman, J., T. Hastie, and R. Tibshirani, The elements of statistical learning, Vol. 1. Springer series in statistics New York, 2001.

[96] Bengio, Y., P. Simard, and P. Frasconi, learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, Vol. 5, No. 2, 1994, pp. 157–166.

[97] LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. Nature, Vol. 521, No. 7553, 2015, pp.436–444.

[98] Hochreiter, S. and J. Schmidhuber, Long short-term memory. Neural computation, Vol. 9, No. 8, 1997, pp. 1735–1780.

[99] Gers, F. A., J. Schmidhuber, and F. Cummins, learning to forget: Continual prediction with LSTM, 1999.

[100] Olah, C., Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015, accessed: 2017-06-30.

[101] Fivethirtyeight, uber-tlc-foil-response. https://tex.stackexchange.com/questions/3587/how-can-i-use-bibtex-to-cite-a-web-page, 2015, accessed: 2017-03-28.

[102] GAIA Open Dataset, https://outreach.didichuxing.com/research/opendata/en/, last accessed 2018/07/21.

[103] DiDi – Wikipedia, https://en.wikipedia.org/wiki/DiDi, last accessed 2018/11/01.

[104] World Weather Online, https://www.worldweatheronline.com/lang/en-us/, last accessed 2018/07/21.

[105] Hou, Y., V. Garikapati, J. Sperling, A. Henao, S. Young. A deep learning approach for TNC trip demand prediction considering spatial-temporal features. 98th Annual Meeting of Transportation Research Board (2019).

[106] Performing Convolution Operations, https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html, last accessed 2018/08/01.

[107] Pooling Layer - Artificial Intelligence, https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/pooling_layer.html, last accessed 2018/08/01.

[108] "Self-Driving Cars Explained," Union of Concerned Scientists. [Online]. Available: https://www.ucsusa.org/clean-vehicles/how-self-driving-cars-work. [Accessed: 3-March-2019].

[109] "10 Advantages of Autonomous Vehicles," ITSdigest. [Online]. Available: https://www.itsdigest.com/10-advantages-autonomous-vehicles. [Accessed: 3-March-2019].

[110] N. J. Goodall, "Vehicle automation and the duty to act", Proceedings of the Twenty-first World Congress on Intelligent Transport Systems, 2014.

[111] E. Kaplan, and C. Hegarty, "Understanding GPS: principles and applications", Artech house, 2005.

[112] N. M. Drawil, H. M. Amar and O. A. Basir, "GPS Localization Accuracy Classification: A Context-Based Approach," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 262-273, March 2013.

[113] A. Shetty, G. Gao, “Covariance estimation for gps-lidar sensor fusion for uavs,” Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017), September 2017.

[114] E. J. Krakiwsky, C. B. Harris and R. V. C. Wong, "A Kalman filter for integrating dead reckoning, map matching and GPS positioning," *IEEE PLANS '88.,Position Location and Navigation Symposium, Record. 'Navigation into the 21st Century'.*, Orlando, FL, USA, 1988, pp. 39-46.

[115] A. Soloviev and D. Venable, "Integration of GPS and vision measurements for navigation in GPS challenged environments," IEEE/ION Position, Location and Navigation Symposium, Indian Wells, CA, 2010, pp. 826-833.

[116] C. R. Jung and C. R. Kelber, "A lane departure warning system using lateral offset with uncalibrated camera," *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, Vienna, 2005, pp. 102-107.

[117] Xiangjing An, Mo Wu and Hangen He, "A Novel Approach to Provide Lane Departure Warning Using Only One Forward-Looking Camera," *International Symposium on Collaborative Technologies and Systems (CTS'06)*, Las Vegas, NV, 2006, pp. 356-362.

[118] J. Liu, Y. Su, M. Ko and P. Yu, "Development of a Vision-Based Driver Assistance System with Lane Departure Warning and Forward Collision Warning Functions," *2008 Digital Image Computing: Techniques and Applications*, Canberra, ACT, 2008, pp. 480-485.

[119] I. Baek, M.He, “Vehicles Lane-changing Behavior Detection,” arXiv preprint arXiv:1808.07518, 2018.

[120] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning", *Nature*, vol. 521, pp. 436-444, May 2015.

[121] Z. Wang, X. Wang, L. Zhao and G. Zhang, “Vision-Based Lane Departure Detection Using a Stacked Sparse Autoencoder,” *Mathematical Problems in Engineering*, vol. 2018, 2018.

[122] A. Gurghian, T. Alexandru, S. V. Bailur, K. J. Carey, and V. N. Murali. "Deeplanes: End-to-end lane position estimation using deep neural networksa." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 38-45. 2016.

[123] M. Mori, C. Miyajima, T. Hirayama, N. Kitaoka and K. Takeda, "Integrated modeling of driver gaze and vehicle operation behavior to estimate risk level during lane changes," *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, The Hague, 2013, pp. 2020-2025.

[124] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778.

[125] R.P. Monti, S. Tootoonian, and R. Cao, "Avoiding degradation in deep feed-forward networks by phasing out skip-connections," *International Conference on Artificial Neural Networks*, 2018, Springer, Cham, pp. 447-456.

[126] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization," ArXiv E-Prints 1412, arXiv:1412.6980, 2014

[127] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "TensorFlow: A System for Large-Scale Machine Learning.," *Symposium on Operating Systems Design and Implementation*, vol. 16, pp. 265–283, 2016.

[128] A. Alekseenko *et al*., "ITS+DM Hackathon (ITSC 2017): Lane Departure Prediction with Naturalistic Driving Data," in *IEEE Intelligent Transportation Systems Magazine*.

[129] T. Chen, C. Guestrin, "Xgboost: A scalable tree boosting system", *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, pp. 785-794, 2016.

[130] Experiment: How Fast Your Brain Reacts to Stimuli", *Backyardbrains.com*, 2019. [Online]. Available: https://backyardbrains.com/experiments/reactiontime. [Accessed: 15- Apr- 2019].

[131] Milanés, V., Shladover, S., Spring, J., Nowakowski, C., Kawazoe, H., and Nakamura, M. (2014). Cooperative Adaptive Cruise Control in Real Traffic Situations, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 15, No. 1, pp. 296 – 305.

[132] Talebpour, A., Mahmassani, H., and Hamdar, S. (2015). Modeling Lane-Changing Behavior in a Connected Environment: A Game Theory Approach. *Transportation Research Procedia*, Vol. 7, pp. 420 – 440.

[133] Xie, Y., Zhang, H., Gartner, N. and Arsava, T. (2017). Collaborative Merging Strategy for Freeway Ramp Operations in a Connected and Autonomous Vehicles Environment, *Journal of Intelligent Transportation Systems*, 21:2, 136-147, DOI: 10.1080/15472450.2016.1248288

[134] Hao, P., Wu, G., Boriboonsomsin, K., and Barth, M. (2018). Eco-Approach and Departure (EAD) application for actuated signals in real-world traffic. *IEEE Transactions on Intelligent Transportation Systems*, 20 (1), 30-40.

[135] Nowakowski, C., O'Connell, J., Shladover, S., and Cody, D. (2010). Cooperative Adaptive Cruise Control: Driver Acceptance of Following Gap Settings Less Than One Second. *Proceedings of the Human Factors and Ergonomics Society 54th Annual Meeting*. Santa Monica, CA: The Human Factors and Ergonomics Society.

[136] Wang, Z., Wu, G., and Barth, M. J. (2018). A Review on Cooperative Adaptive Cruise Control (CACC) Systems: Architectures, Controls, and Applications. *21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, pp. 2884-2891.

[137] Shladover, S., Lu, X., Yang, S., Ramezani, H., Spring, J., Nowakowski, C., Nelson, D., Thompson, D., Kailas, A., and McAuliffe, B. (2018). Cooperative Adaptive Cruise Control

(CACC) For Partially Automated Truck Platooning: Final Report. Federal Highway Administration – Exploratory Advanced Research Program.

[138] U.S. Department of Transportation. (2016) Applications for the Environment: Real-Time Information Synthesis (AERIS). http://www.its.dot.gov/aeris/, Accessed 01 March 2019.

[139] Shaheen, S., Chan, N., Bansal, A., and Cohen, A. (2015). "Shared Mobility: Definitions, Industry Developments, and Early Understanding". White Paper, November

[140] Martin E., and Shaheen, S. (2011). The Impact of Carsharing on Public Transit and Non-Motorized Travel. *Energies* 2011, 4, 2094-2114.

[141] Shaheen, S. and Cohen, A. (2012). Innovative Mobility Carsharing Outlook. Transportation Sustainability Research Center (TSRC), UC Berkeley, Fall 2012.

[142] Viechnicki, P., Khuperkar, A., Fishman, T., and Eggers, W. (2015). Smart Mobility: Reducing Congestion and Fostering Faster, Greener, and Cheaper Transportation Options. *Deloitte University Press*, May 2015.

[143] Zmud, J., Williams, T., Outwater, M., Bradley, M., Kalra, Nidhi, and Row, S. (2018). Updating Regional Transportation Planning and Modeling Tools to Address Impacts of Connected and Automated Vehicles, Volume 2: Guidance. NCHRP Research Report 896.

[144] Harb, M., Xiao, Y., Circella, G., Mokhtarian, P. and Walker, J., (2018). Projecting Travelers into a World of Self-driving Vehicles: Estimating Travel Behavior Implications via a Naturalistic Experiment, *Transportation* (2018) 45: 1671.

[145] Clewlow, R., and Mishra, G. (2017). Disruptive Transportation: The Adoption, Utilization, and Impacts of Ride-Hailing in the United States. NCST Research Report, UCD-ITS-RR-17-07.

[146] Circella, G., Alemi, F., Tiedeman, K, Handy, S. and Mokhtarian, P. (2018). The Adoption of Shared Mobility in California and Its Relationship with Other Components of Travel Behavior, NCST Research Report.

[147] Wenzel, T., Rames, C., Kontou, E., and Henao, A. (2019). Travel and energy implications of ridesourcing service in Austin, Texas. *Transportation Research Part D: Transport and Environment*, 70, 18-34.

[148] Hao, P., Boriboonsomsin, K., Wang, C., Wu, G., and Barth, M. (2018). Connected Eco-Approach and Departure (EAD) system for diesel trucks. Proceedings of the 97th Annual Meeting of Transportation Research Board, Washington, DC.

[149] Hao, P., Boriboonsomsin, K., Wu, G., Gao, Z., LaClair, T, and Barth, M. (2019). Deeply Integrated Vehicle Dynamic and Powertrain Operation for Efficient Plug-in Hybrid Electric Bus. Proceedings of the 98th Annual Meeting of Transportation Research Board, Washington, DC.

[150] Martin-Gasulla, M., Sukennik, P., Lohmiller, M., 2019. Investigation of the Impact on Throughput of Connected Autonomous Vehicles with Headway Based on Leading Vehicle Type, Proceeding of the 98th Transportation Research Board, Washington, DC.

[151] Liu, H. et al, 2018. Using Cooperative Adaptive Cruise Control (CACC) to Form High-Performance Vehicle Streams—Final Report, DTFH61-13-H-00013, FHWA-EAR.

[152] Goudy, R. (2018). Traffic Optimization for Signalized Corridors (TOSCo). Invited talk at ARPA-E NEXTCAR 2018 Annual Meeting. https://arpa-e.energy.gov/sites/default/files/Goudy_NEXTCAR_TOSCo%20Overview_Final.pdf. Accessed on July 31, 2019.

[153] MATSim, https://www.matsim.org, Accessed on July 31, 2019.

[154] POLARIS Transportation System Simulation Tool, https://www.anl.gov/es/polaris-transportation-system-simulation-tool, Accessed on July 31, 2019.

[155] BEAM, beam.lbl.gov, Accessed on July 31, 2019.

[156] SCAG, https://www.scag.ca.gov/DataAndTools/Pages/TransportationModels.aspx, Accessed on July 31, 2019.

[157] Shabanpour, R., Auld, J., Mohammadian, A. K., and Stephens, T. S. (2017). Developing a Platform to Analyze Behavioral Impacts of Connected Automated Vehicles at the National Level. Proceedings of the 96th Annual Meeting of Transportation Research Board, Washington, DC.

[158] National Household Travel Survey, https://nhts.ornl.gov/. Accessed on July 31, 2019.